Marco Dorigo et al. (Eds.)

LNCS 6234

Swarm Intelligence

7th International Conference, ANTS 2010
Brussels, Belgium, September 2010
Proceedings

Springer

Lecture Notes in Computer Science 6234

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Marco Dorigo Mauro Birattari
Gianni A. Di Caro René Doursat
Andries P. Engelbrecht Dario Floreano
Luca Maria Gambardella Roderich Groß
Erol Şahin Hiroki Sayama
Thomas Stützle (Eds.)

Swarm Intelligence

7th International Conference, ANTS 2010
Brussels, Belgium, September 8-10, 2010
Proceedings

Springer

Volume Editors

Marco Dorigo, E-mail: mdorigo@ulb.ac.be
Mauro Birattari, E-mail: mbiro@ulb.ac.be
Gianni A. Di Caro, E-mail: gianni@idsia.ch
René Doursat, E-mail: rene.doursat@polytechnique.edu
Andries P. Engelbrecht, E-mail: engel@cs.up.ac.za
Dario Floreano, E-mail: dario.floreano@epfl.ch
Luca Maria Gambardella, E-mail: luca@idsia.ch
Roderich Groß, E-mail: r.gross@sheffield.ac.uk
Erol Şahin, E-mail: erol@ceng.metu.edu.tr
Hiroki Sayama, E-mail: sayama@binghamton.edu
Thomas Stützle, E-mail: stuetzle@ulb.ac.be

Library of Congress Control Number: 2010933078

CR Subject Classification (1998): I.2, F.1, F.2, H.4, C.2, H.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-642-15460-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-15460-7 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

These proceedings contain the papers presented at ANTS 2010, the 7th International Conference on Swarm Intelligence, organized by IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium, during September 8–10, 2010. The ANTS series started in 1998 with the First International Workshop on Ant Colony Optimization (ANTS 1998), which attracted more than 50 participants. Since then ANTS, which is held bi-annually, has gradually become an international forum for researchers in the wider field of swarm intelligence. In the past (since 2004), this development has been acknowledged by the inclusion of the term "Swarm Intelligence" (next to "Ant Colony Optimization") in the conference title. This year's ANTS conference was officially devoted to the field of swarm intelligence as a whole, without any bias towards specific research directions. As a result, the title of the conference was changed to "The International Conference on Swarm Intelligence." This name change is already in place this year, and future ANTS conferences will continue to use the new title.

This volume contains the best papers selected out of 99 submissions. Of these, 28 were accepted as full-length papers, while 27 were accepted as short papers. This corresponds to an overall acceptance rate of 56%. Also included in this volume are 14 extended abstracts.

Of the full-length papers, 15 were selected for oral presentation at the conference. All other contributions, including short papers and extended abstracts, were presented in the form of poster presentations. Following the conference, the journal *Swarm Intelligence* will publish extended versions of some of the best papers presented at the conference.

The conference featured three distinguished plenary talks: "Locating and Tracking Multiple Optima Using Particle Swarm Optimization" by Andries Engelbrecht, "Emergent Coordination in Fish Schools and Human Crowds" by Guy Theraulaz, and "Self-Reconfigurable Robots, Digital Hormones, and Swarm Morphallaxis" by Wei-Mei Shen. A special session, jointly organized by René Doursat and Hiroki Sayama, focused on recent developments in the area of morphogenetic engineering. A workshop organized by Dario Floreano provided opportunities to discuss research challenges related to the EU project Swarmanoid.

We take this opportunity to thank the large number of people that were involved in making this conference a success. We express our gratitude to the authors who contributed their work, to the members of the International Programme Committee, to the additional referees for their qualified and detailed reviews, and to the people at IRIDIA for helping with organizational matters. We thank the keynote speakers for their inspiring talks. Finally, we thank our sponsors: AntOptima, the Belgian Fund for Scientific Research-FNRS, the European Coordinating Committee for Artificial Intelligence, the French Community of Belgium, the IEEE Computational Intelligence Society, and Wolfram Research.

We hope the reader will find this volume useful both as a reference to current research in swarm intelligence and as a starting point for future work.

July 2010

Marco Dorigo
Mauro Birattari
Gianni A. Di Caro
René Doursat
Andries P. Engelbrecht
Dario Floreano
Luca Maria Gambardella
Roderich Groß
Erol Şahin
Hiroki Sayama
Thomas Stützle

Organization

ANTS 2010 was organized by IRIDIA, CoDE, Université Libre de Bruxelles, Belgium.

General Chair

Marco Dorigo	Université Libre de Bruxelles, Belgium

Technical Program Chairs

Gianni A. Di Caro	IDSIA, USI-SUPSI, Switzerland
Andries P. Engelbrecht	University of Pretoria, South Africa
Luca Maria Gambardella	IDSIA, USI-SUPSI, Switzerland
Erol Şahin	Middle East Technical University, Turkey

Chairs of the Special Session on Morphogenetic Engineering

René Doursat	ISC-PIF, France
Hiroki Sayama	Binghamton University, NY, USA

Chair of the Co-located Workshop on Swarmanoid

Dario Floreano	EPFL, Switzerland

Publication Chair

Roderich Groß	The University of Sheffield, UK

Organization Chairs

Mauro Birattari	Université Libre de Bruxelles, Belgium
Thomas Stützle	Université Libre de Bruxelles, Belgium

Publicity Chair

Xiaodong Li	RMIT University, Australia

Local Arrangements

Manuele Brambilla	Université Libre de Bruxelles, Belgium

Program Committee

Andy Adamatzky	University of the West of England, UK
Paul Andrews	University of York, UK
Daniel Angus	University of Queensland, Australia
Tucker Balch	Georgia Institute of Technology, GA, USA
Julio R. Banga	CSIC, Spain
Wolfgang Banzhaf	Memorial University of Newfoundland, Canada
Jacob Beal	BBN Technologies, MA, USA
Gerardo Beni	University of California, CA, USA
Cyrille Bertelle	Université de Havre, France
Tim Blackwell	Goldsmiths, University of London, UK
Christian Blum	Universitat Politècnica de Catalunya, Spain
Vivek Borkar	Tata Institute of Fundamental Research, India
Fernando Buarque	Universidade de Pernambuco, Brazil
Supiya Charoensiriwath	NECTEC, Thailand
Marco Chiarandini	University of Southern Denmark, Denmark
Anders L. Christensen	Instituto Universitario de Lisboa, Portugal
Maurice Clerc	University of Essex, UK
Leandro Coelho	Pontifícia Universidade Católica do Paraná, Brazil
Carlos Coello Coello	CINVESTAV-IPN, Mexico
Oscar Cordón	European Centre for Soft Computing, Spain
Swagatam Das	Jadavpur University, India
Prithviraj Raj Dasgupta	University of Nebraska, NE, USA
Kusum Deep	Indian Institute of Technology Roorkee, India
Karl Doerner	Universität Wien & Salzburg Research, Austria
Hai-Bin Duan	Beihang University, China
Frederick Ducatelle	IDSIA, USI-SUPSI, Switzerland
Mohammed El-Abd	University of Waterloo, Canada
Susana Esquivel	Universidad Nacional de San Luis, Argentina
Nazim Fatès	INRIA, France
Juan L. Fernández-Martínez	Universidad de Oviedo, Spain
Jonathan Fieldsend	Exeter University, UK
Simon Garnier	Princeton University, NJ, USA
Veysel Gazi	Ekonomi ve Teknoloji Universitesi, Turkey
Marde Greeff	University of Pretoria, South Africa
Julie Greensmith	University of Nottingham, UK
Frédéric Guinand	Université du Havre, France
Walter Gutjahr	Universität Wien, Austria
Saman Halgamuge	Melbourne School of Engineering, Australia

Julia Handl	University of Manchester, UK
Emma Hart	Edinburgh Napier University, UK
Richard Hartl	Universität Wien, Austria
Poul Heegaard	NTNU, Norway
Tim Hendtlass	Swinburne University of Technology, Australia
Holger Hoos	University of British Columbia, Canada
Ani Hsieh	Drexel University, PA, USA
Thomas Jansen	University College Cork, Ireland
Mark Jelasity	Szegedi Tudomanyegyete, Hungary
Yaochu Jin	University of Surrey, UK
Alexander John	Universität zu Köln, Germany
Krishnanand Kaipa	University of Vermont, VT, USA
James Kennedy	Bureau of Labor Statistics, DC, USA
Serge Kernbach	Universität Stuttgart, Germany
Joshua Knowles	University of Manchester, UK
Oliver Korb	Cambridge Crystallographic Data Centre, UK
Pietro Liò	University of Cambridge, UK
Manuel López-Ibáñez	Université Libre de Bruxelles, Belgium
Katherine Malan	University of Pretoria, South Africa
Vittorio Maniezzo	Università di Bologna, Italy
Alcherio Martinoli	EPFL, Switzerland
Ronaldo Menezes	Florida Institute of Technology, FL, USA
Daniel Merkle	University of Southern Denmark, Denmark
Bernd Meyer	Monash University, Australia
Olivier Michel	Université Paris XII, France
Martin Middendorf	Universität Leipzig, Germany
Chilukuri Mohan	Syracuse University, NY, USA
Francesco Mondada	EPFL, Switzerland
Nicolas Monmarché	Université de Tours, France
Sara Montagna	Università di Bologna, Italy
Roberto Montemanni	IDSIA, USI-SUPSI, Switzerland
Marco A. Montes De Oca	Université Libre de Bruxelles, Belgium
Sanaz Mostaghim	Karlsruher Institut für Technologie, Germany
Frank Neumann	Max-Planck-Institut für Informatik, Germany
Giuseppe Nicosia	Università di Catania, Italy
Fernando Nino	National University of Colombia, Colombia
Ann Nowé	Vrije Universiteit Brussel, Belgium
Mahamed Omran	Gulf University for Science and Technology, Kuwait
Lisa Osadciw	Syracuse University, NY, USA
Ender Özcan	University of Nottingham, UK
Lynne E. Parker	University of Tennessee, TN, USA
Rafael Stubs Parpinelli	Universidade do Estado de Santa Catarina, Brazil
Kostantinos Parsopoulos	University of Ioannina, Greece

Van Dyke Parunak	NewVectors division of TTGSI, MI, USA
Paola Pellegrini	Università degli Studi di Trieste, Italy
Gilbert Peterson	Air Force Institute of Technology, OH, USA
Jim Pugh	EPFL, Switzerland
Marc Reimann	University of Warwick, UK
Aristides Requicha	University of Southern California, CA, USA
Andrea Roli	Università di Bologna, Italy
Biswanath Samanta	Villanova University, PA, USA
Michael Sampels	Université Libre de Bruxelles, Belgium
Thomas Schmickl	Karl-Franzens-Universität Graz, Austria
Giovanni Sebastiani	IAC "M. Picone", Italy
Kevin Seppi	Brigham Young University, UT, USA
Christine Solnon	Université Claude Bernard, France
William M. Spears	University of Wyoming, WY, USA
Antoine Spicher	Université Paris XII, France
Thomas Stibor	Technische Universität München, Germany
Kasper Støy	University of Southern Denmark, Denmark
Ponnuthurai Suganthan	Nanyang Technological University, Singapore
El-Ghazali Talbi	Université de Lille, France
Guy Theraulaz	Université Paul Sabatier, France
Jon Timmis	University of York, UK
Kohji Tomita	AIST, Japan
Ioan Cristian Trelea	AgroParisTech, France
Vito Trianni	ISTC-CNR, Italy
Elio Tuci	ISTC-CNR, Italy
Ali Emre Turgut	Université Libre de Bruxelles, Belgium
Supiya Ujjin	University College London, UK
Richard T. Vaughan	Simon Fraser University, Canada
Kalyan Veeramachaneni	Syracuse University, NY, USA
Ganesh K. Venayahamoorthy	Missouri University of Science and Technology, MO, USA
Mario Ventresca	University of Waterloo, Canada
Michael Vrahatis	University of Patras, Greece
Justin Werfel	New England Complex Systems Inst., MA, USA
Alan F.T. Winfield	University of the West of England, UK
Carsten Witt	Technical University of Denmark, Denmark
Jun Zhang	Sun Yat-sen University, China

Additional Referees

Stefano Benedettini	Stefano Nolfi	James Styles
Arne Brutschy	Rehan O'Grady	Markus Waibel
Chris Fawcett	Andres Perez-Uribe	Steffen Wischmann
Frank Hutter	Carlo Pinciroli	Xiao-Feng Xie
Nithin Mathews	Onur Soysal	
Sara Mitri	Valerio Sperati	

Sponsoring Institutions

AntOptima, Lugano, Switzerland
http://www.antoptima.com

Belgian Fund for Scientific Research–FNRS
http://www.fnrs.be

European Coordinating Committee for Artificial Intelligence
http://www.eccai.org

French Community of Belgium (through the research project META-X)
http://www.cfwb.be

IEEE Computational Intelligence Society (as a technical co-sponsor)
http://www.ieee-cis.org

Wolfram Research
http://www.wolfram.com/

Table of Contents

A Graph-Based Developmental Swarm Representation and Algorithm 1
Sebastian von Mammen, David Phillips, Timothy Davison, and Christian Jacob

A Modified Particle Swarm Optimization Algorithm for the Best Low Multilinear Rank Approximation of Higher-Order Tensors 13
Pierre B. Borckmans, Mariya Ishteva, and Pierre-Antoine Absil

A Robotic Validation of the Attractive Field Model: An Inter-disciplinary Model of Self-regulatory Social Systems 24
Md. Omar Faruque Sarker and Torbjørn S. Dahl

A Thermodynamic Approach to the Analysis of Multi-robot Cooperative Localization under Independent Errors 36
Yotam Elor and Alfred M. Bruckstein

An Alternative $ACO_{\mathbb{R}}$ Algorithm for Continuous Optimization Problems 48
Guillermo Leguizamón and Carlos A. Coello Coello

An Efficient Optimization Method for Revealing Local Optima of Projection Pursuit Indices 60
Souad Larabi Marie-Sainte, Alain Berro, and Anne Ruiz-Gazen

Ant Colony Optimisation for Ligand Docking 72
Oliver Korb and Jason Cole

Antbots: A Feasible Visual Emulation of Pheromone Trails for Swarm Robots 84
Ralf Mayet, Jonathan Roberz, Thomas Schmickl, and Karl Crailsheim

Automatic Configuration of Multi-Objective ACO Algorithms 95
Manuel López-Ibáñez and Thomas Stützle

Autonomous Morphogenesis in Self-assembling Robots Using IR-Based Sensing and Local Communications 107
Wenguo Liu and Alan F.T. Winfield

Autonomous Multi-agent Cycle Based Patrolling 119
Yotam Elor and Alfred M. Bruckstein

Biologically Realistic Primitives for Engineered Morphogenesis 131
Justin Werfel

Evaluating the Robustness of Activator-Inhibitor Models for Cluster Head Computation 143
Lidia Yamamoto and Daniele Miorandi

Evolution of Self-organised Path Formation in a Swarm of Robots 155
Valerio Sperati, Vito Trianni, and Stefano Nolfi

Extensions to the Ant-Miner Classification Rule Discovery Algorithm . . . 167
Khalid M. Salama and Ashraf M. Abdelbar

Functional Blueprints: An Approach to Modularity in Grown Systems 179
Jacob Beal

Heterogeneous Particle Swarm Optimization 191
Andries P. Engelbrecht

Modern Continuous Optimization Algorithms for Tuning Real and Integer Algorithm Parameters 203
Zhi Yuan, Marco A. Montes de Oca, Mauro Birattari, and Thomas Stützle

Multi-agent Deployment on a Ring Graph 215
Yotam Elor and Alfred M. Bruckstein

Multi-Swarm Optimization for Dynamic Combinatorial Problems: A Case Study on Dynamic Vehicle Routing Problem 227
Mostepha Redouane Khouadjia, Enrique Alba, Laetitia Jourdan, and El-Ghazali Talbi

Off-line *vs.* On-line Tuning: A Study on $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for the TSP 239
Paola Pellegrini, Thomas Stützle, and Mauro Birattari

Opinion Dynamics for Decentralized Decision-Making in a Robot Swarm 251
Marco A. Montes de Oca, Eliseo Ferrante, Nithin Mathews, Mauro Birattari, and Marco Dorigo

Positional Communication and Private Information in Honeybee Foraging Models 263
Peter Bailis, Radhika Nagpal, and Justin Werfel

Rank Based Particle Swarm Optimization 275
Affan Khan, Muhammad Sadeequllah, Riaz-ul-Hasnain, and Azzam-ul-Asar

Self-organized Task Partitioning in a Swarm of Robots 287
Marco Frison, Nam-Luc Tran, Nadir Baiboun, Arne Brutschy, Giovanni Pini, Andrea Roli, Marco Dorigo, and Mauro Birattari

Slime Mold Inspired Path Formation Protocol for Wireless Sensor Networks 299
Ke Li, Kyle Thomas, Claudio Torres, Louis Rossi, and Chien-Chung Shen

Solving the Multi-dimensional Multi-choice Knapsack Problem with the Help of Ants 312
Shahrear Iqbal, Md. Faizul Bari, and M. Sohel Rahman

Theoretical Properties of Two ACO Approaches for the Traveling Salesman Problem 324
Timo Kötzing, Frank Neumann, Heiko Röglin, and Carsten Witt

Short Papers

A Cooperative Network Game Efficiently Solved via an Ant Colony Optimization Approach 336
Pablo Romero, Franco Robledo, Pablo Rodríguez-Bocca, Darío Padula, and María Elisa Bertinat

A Deterministic Metaheuristic Approach Using "Logistic Ants" for Combinatorial Optimization 344
Rodolphe Charrier, Christine Bourjot, and François Charpillet

A Model Based Ant Colony Design for the Protein Engineering Problem 352
Matteo Borrotti, Davide De Lucrezia, Giovanni Minervini, and Irene Poli

ACOPHY: A Simple and General Ant Colony Optimization Approach for Phylogenetic Tree Reconstruction 360
Huy Q. Dinh, Bui Quang Minh, Hoang Xuan Huan, and Arndt von Haeseler

ACS Searching for D_{4t}-Hadamard Matrices 368
Víctor Álvarez, José Andrés Armario, María Dolores Frau, Félix Gudiel, Belén Güemes, Elena Martín, and Amparo Osuna

Ant Based Semi-supervised Classification 376
Anindya Halder, Susmita Ghosh, and Ashish Ghosh

Automatic Generation of Optimised Working Time Models in Personnel Planning 384
Volker Nissen and Maik Günther

Bee-Sensor: A Step Towards Meta-Routing Strategies in Hybrid Ad Hoc Networks 392
Israr Ullah, Muhammad Saleem, and Muddassar Farooq

Cooperation in a Heterogeneous Robot Swarm through Spatially Targeted Communication 400
Nithin Mathews, Anders Lyhne Christensen, Rehan O'Grady, and Marco Dorigo

Early-Stage Diagnosis of Endogenous Diseases by Swarms of Nanobots: An Applicative Scenario 408
Paolo Amato, Massimo Masserini, Giancarlo Mauri, and Gianfranco Cerofolini

EDA-PSO: A Hybrid Paradigm Combining Estimation of Distribution Algorithms and Particle Swarm Optimization 416
Endika Bengoetxea and Pedro Larrañaga

Emergent Flocking with Low-End Swarm Robots 424
Christoph Moeslinger, Thomas Schmickl, and Karl Crailsheim

Exploiting Loose Horizontal Coupling in Evolutionary Swarm Robotics 432
Jennifer Owen, Susan Stepney, Jonathan Timmis, and Alan F.T. Winfield

Formal Verification of Probabilistic Swarm Behaviours 440
Savas Konur, Clare Dixon, and Michael Fisher

Inverse Modeling in Geoenvironmental Engineering Using a Novel Particle Swarm Optimization Algorithm 448
Tadikonda Venkata Bharat and Jitendra Sharma

Mobile Stigmergic Markers for Navigation in a Heterogeneous Robotic Swarm 456
Frederick Ducatelle, Gianni A. Di Caro, Alexander Förster, and Luca Gambardella

Motif Finding Using Ant Colony Optimization 464
Salim Bouamama, Abdellah Boukerram, and Amer F. Al-Badarneh

Multiple Ant Colony System for Substructure Discovery 472
Oscar Cordón, Arnaud Quirin, and Rocío Romero-Zaliz

Opportunistic Ant-Based Path Management for Wireless Mesh Networks 480
Laurent Paquereau and Bjarne E. Helvik

Parallel Ant Colony Optimization Algorithm on a Multi-core Processor 488
Shigeyoshi Tsutsui and Noriyuki Fujimoto

Particle Swarm Optimization in High Dimensional Spaces 496
Juan L. Fernández-Martínez, Tapan Mukerji, and Esperanza García-Gonzalo

Particle Swarm Optimization of Bollinger Bands 504
Matthew Butler and Dimitar Kazakov

Protein Structure Prediction in Lattice Models with Particle Swarm Optimization 512
Andrei Băutu and Henri Luchian

Short and Robust Communication Paths in Dynamic Wireless Networks 520
Yoann Pigné and Frédéric Guinand

The ACO Encoding 528
Alberto Moraglio, Fernando E.B. Otero, and Colin G. Johnson

The Complexity of Grid Coverage by Swarm Robotics 536
Yaniv Altshuler and Alfred M. Bruckstein

The Design of an Active Structural Vibration Reduction System Using a Modified Particle Swarm Optimization 544
Adam Schmidt

Extended Abstracts

Ant Colony Extended: Search in Solution Spaces with a Countably Infinite Number of Solutions 552
Jose B. Escario, Juan F. Jimenez, and Jose M. Giron-Sierra

Automatic Parameter Configuration of Particle Swarm Optimization by Classification of Function Features 554
Tjorben Bogon, Georgios Poursanidis, Andreas D. Lattner, and Ingo J. Timm

Constructing Low-Cost Swarm Robots That March in Column Formation 556
Asuki Kouno, Shigeru Takano, and Einoshin Suzuki

Coordinating Heterogeneous Swarms through Minimal Communication among Homogeneous Sub-swarms 558
Carlo Pinciroli, Rehan O'Grady, Anders Lyhne Christensen, and Marco Dorigo

Effect of Particle Initialization on the Performance of Particle Swarm Niching Algorithms 560
Isabella Schoeman and Andries P. Engelbrecht

Energy Efficient Swarm Deployment for Search in Unknown Environments 562
Timothy Stirling and Dario Floreano

Genetic Encoding of Robot Metamorphosis: How to Evolve a Glider with a Genetic Regulatory Network 564
Anne C. van Rossum

How Ant Systems Can Help in Management of pH for Industrial Wastewater Discharges 566
Marta Verdaguer, Jordi Giró, Narcís Clara, and Manel Poch

Hybrid Metaheuristic Combining Ant Colony Optimization and H-Method 568
Leonid Hulianytskyi and Sergii Sirenko

Increasing Individual Density Reduces Extra-variance in Swarm Intelligence 570
Ryusuke Fujisawa, Shigeto Dobata, and Fumitoshi Matsuno

"Look out!": Socially-Mediated Obstacle Avoidance in Collective Transport 572
Eliseo Ferrante, Manuele Brambilla, Mauro Birattari, and Marco Dorigo

On Possible Connections between Ant Algorithms and Random Matrix Theory 574
Carlo Mastroianni

Soft Variable Fixing in Path Relinking: An Application to ACO Codes 576
Antonio Bolufé Röhler, Marco A. Boschetti, and Vittorio Maniezzo

Training Randomly Connected, Recurrent Artificial Neural Networks Using PSO 578
Vytautas Jancauskas

Author Index 581

A Graph-Based Developmental Swarm Representation and Algorithm

Sebastian von Mammen[1], David Phillips[1],
Timothy Davison[1], and Christian Jacob[1,2]

[1] Dept. of Computer Science, University of Calgary, Canada
s.vonmammen@ucalgary.ca
[2] Dept. of Biochemistry and Molecular Biology, University of Calgary, Canada

Abstract. Modelling natural processes requires the implementation of an expressive representation of the involved entities and their interactions. We present *swarm graph grammars* (SGGs) as a bio-inspired modelling framework that integrates aspects of formal grammars, graph-based representation and multi-agent simulation. In SGGs, the substitution of subgraphs that represent locally defined agent interactions drive the computational process of the simulation. The generative character of formal grammars is translated into an agent's *metabolic* interactions, i.e. creating or removing agents from the system. Utilizing graphs to describe interactions and relationships between pairs or sets of agents offers an easily accessible way of modelling biological phenomena. Property graphs emerge through the application of local interaction rules; we use these graphs to capture various aspects of the interaction dynamics at any given step of a simulation.

1 Introduction

We are interested in modelling complex biological systems at various levels of scale, i.e. from the biomolecular level [33] to cells [13] to systems [15], etc. Different levels of resolution often require different computational techniques, such as differential equation solvers to compute physics or fluid dynamics, or engines that execute high-level agent behaviours that implement rich interaction policies and complex strategies [39]. Independent of the specific computational approaches that drive the simulation processes, they all rely on state changes, the principle of digital computation. Furthermore, a system's state determines the introduced changes, probabilistically or deterministically. This idea is emphasized in numerous computational representations such as Markov chains [1] or cellular automata [38]. The state of a system is generally understood as the states of all its subsystems including their interrelations. Consequently, states and relations are interchangeable terms that provide the condition for change, or the antecedent for a consequent in a simple *If-then rule*. A set of probabilistic rules (like in Markov chain systems) works well to represent the activities of decentralized, self-organizing swarm agents [3,4,5,15,14], including swarm-based developmental systems [25,24].

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 1–12, 2010.

Rule-based swarm systems seem to be a good fit to capture biological models. However, there are several hurdles that make it hard to deploy swarm models in fields outside of computer science. (1) The predicates and actions that drive the simulations—e.g. the detection of a chemical signal or the deposition of a particle—depend on the modelling domains and are usually re-implemented for different experiments. Still, many of these operations can be abstracted, parametrically adjusted and reused in different contexts. The integration of these operations into a rule-based formalism also makes it possible to utilize functionality from various computational engines such as physics engines or general differential equation solvers within one modelling framework. (2) Depending on the degree of specificity of a rule's condition and its associated actions, a theoretically simple interaction can result in an over-complicated representation. A graphical description of the predicates and the associated actions can amend this issue. (3) As swarm simulations often exhibit complex behaviours, little details—for example the order of execution and the discretization steps in a simulation—can greatly influence the outcome. Therefore, we think it is crucial to design models based on a unified algorithmic scheme.

We have devised *swarm graph grammars* (SGGs) to alleviate some of the challenges discussed above. SGGs provide a graphical, rule-based description language to specify swarm agents and a generalized algorithmic framework for the simulation of complex systems. Fundamental operations such as creation or deletion of programmatic objects, as provided by formal grammars, are part of the SGG syntax. Through SGGs we can capture (metabolic) functions at multiple biological scales, i.e. the processes of secretion and diffusion [37], or consumption/removal and production/construction [20], respectively. As a consequence of the graph-based syntax, SGGs capture the simulation in a global graph at each computational step. Thereby, the continuous re-shaping of an interaction topology of a dynamic system is traced and interdependencies that emerge over the course of a simulation are graphically represented.

The remainder of this paper is organized as follows. In Section 2, immediately relevant work in the respective areas of research is presented. Section 3 details swarm graph grammars (SGGs) and their constituents, i.e. swarm individuals, graph grammatical rules, and a general SGG algorithm. Section 4 shows how the SGG formalism is applied in a step by step manner to retrace a simple boid simulation, wasp nest construction, and directed cell growth and proliferation. We conclude with a summary and an outlook on possible future work.

2 Related Work

Cellular automata (CAs) can be considered the first computational developmental models [28]. CAs revolve around state-based interactions of individuals given a fixed interaction topology. However, in the emerging discipline of computational developmental systems, the focus shifted towards constructive expressiveness and thus overshadowed the idea of individual-based modelling. In this section, we briefly review the emergence of CDMs and demonstrate their reunion with agent-based modelling.

2.1 Complex CDMs

Giavitto et al. summarize several approaches to computational developmental models [10]. The most simple ones are considered to be *dynamical systems* with sets of state variables determining their global states. *Structured dynamical systems* are more complex; they are dynamic systems that can be divided into subsystems. Finally, there are *dynamical systems with dynamical structures*, abbreviated as $(DS)^2$-systems, for instance a "developing multi-cellular organism" [12]. In addition, Giavitto et al. describe developmental models as tuples of topology and formalism. L-systems [22], for instance, describe how individual elements of sequences are substituted in parallel. *Group-based data fields* (GBF) [34], on the other hand, operate on sets of units that are connected with a homogeneous, fixed topology not unlike cellular automata [28]. *Map L-systems* [2], similar to *random boolean networks* (RBNs) [16], promote combinatorial topologies on the interacting, or growing, data structures. There are also formalisms that explicitly integrate the topology of the modelled systems, such as *membrane computing* (MC), or *P systems* [29]. P systems draw their inspiration from membrane structures of cells, neural cells and tissues. In a more generalized fashion, *graph grammars* [9] are a means to integrate topological information into any kind of developmental model. Examples are *multiscale tree graphs* (MTGs) and the *modèle géneral de simulation* (MGS) that represent changes of *topological collections* of units by *transformation paths* on a symbolic notation [11].

2.2 Graph-Based CDMs

Kniemeyer et al. have developed *relational growth grammars* (RGGs) which promise, like MGS, to be a universally applicable representation of CDMs [19]. They use RGGs as extensions of parametric L-systems with object-oriented, rule-based, procedural features. In fact, modelling CDMs by graph grammars, like in RGGs, allows for the expression of all developmental data structures commonly used in the computational sciences: multisets, strings, axial trees, and relational structures (edge-labeled directed graphs). Graph grammar-based CDMs can therefore be considered as a universal modelling language, able to simulate standard L-systems, artificial chemistries and ecological systems alike. Kniemeyer et al. successfully applied the RGG model to grow multi-scale models of plants integrating their structure and function [18], and, recently, to grow architectural models [17]. They also suggested that RGGs could support *agent-based* modelling—by interpreting nodes as agents, edges as inter-agent relations, and by driving their interactions through sub-graph substitutions [21].

Almost 20 years before Kniemeyer presented RGGs, Culik et al. had extended L-systems with the means to describe plants through graph structures and their growth through graph grammatical substitutions, which were later on referred to as *graph L-systems* [6]. Shortly afterwards, Nagl investigated the relationship between graph grammars and graph L-systems, concluding that graph grammars can be reduced to graph L-systems and vice versa [27]: identical graphs can be achieved by either sequential graph grammar productions or by parallel

subgraph substitutions as realized in graph L-systems. About another decade later, Lindenmayer argued that relying on maps instead of graphs bears many advantages, e.g. a clear method for mapping between the abstract representation and the natural, growing structures and better performance due to the avoidance of transformations of the representations [23].

Recently, Tomita et al. have presented *graph rewriting automata* [36], in which lattice-based CAs evolve into complex networks through the application of production rules that change local connectivities. Sayama et al. went one step further and considered the local states of a CA to inform the development of *generative network automata* (GNA) [32].

2.3 Swarm-Based CDMs

Developmental systems can be simulated by means of agent-based, decentralized models that incorporate diffusion of molecular signals paired with particular protein or cell behaviours [31]. A generic formalism for agent-based models was provided by Denzinger et al. [7,8] in which an agent is represented as a quadruple $Ag = (Sit, Act, Dat, f_{Ag})$. An agent Ag can find itself in any of the situations expressed in Sit. It can perform the actions described by the set Act. Its internal data areas, i.e. local variables or memory cells, are determined by the set of possible values Dat. Based on the perceived situation and its internal data values, the agent determines the next action through a decision function $f_{Ag} : Sit \times Dat \rightarrow Act$. This representation is very expressive and follows the descriptive methodology of many natural sciences in which the principle of local cause and effect leads to associated emergent phenomena of interest.

Based on these ideas, we have introduced *swarm grammars* (SGs) that merged L-systems with an agent-based modelling approach [24]. In swarm grammars, decentralized swarm agents, or individuals, have the ability to perceive and act in accordance with Denzinger et al.'s agent definition. In particular, SG individuals can react to their local environment, differentiate, reproduce, and create structures by depositing construction elements. Albeit the fact that SGs merge several instrumental biological concepts of developmental, non-linear interaction systems, they do not provide a unified, easy-to-use representation and algorithm that allows for systematic deployment in other scientific disciplines (as discussed in Section 1).

3 Swarm Graph Grammars

We present *swarm graph grammars* as a unified modelling and simulation framework for swarm-based systems that addresses the challenges outlined in Section 1, and provides a unified, graphical, rule-based modelling language for swarm individuals and a generalized simulation algorithm. The graphical description renders model dynamics more tangible and translates local interactions into global, continuously changing interaction networks. We believe that investigations into the development of these networks, in turn, could reveal quantifiable measures

about *emergent* global phenomena. We address Lindenmayer's concerns about the inefficiency of graph-based CDMs by a minimalist subgraph matching procedure that only considers star networks of depth 1 around the corresponding, active reference agent.

3.1 Representation

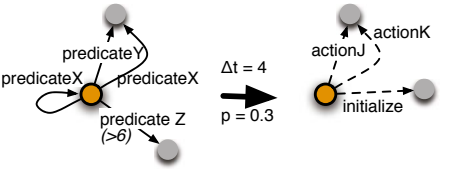


Fig. 1. An SGG rule that queries the reference node itself, other individuals and sets of interaction candidates, to interact with them, delete some and to initialize a new node

An SGG agent's behaviour is described by a set of rules (Figure 1). Each rule tests a set of predicates (solid edges on the left-hand side) and executes a set of actions (dashed edges on the right-hand side) in respect to the acting agent itself (reference node) or other agents. Nodes represent individual agents or sets of agents. In Figure 1, the acting agent is displayed as an orange node with a black border. Other agents or agent groups are depicted as grey nodes. The application of the rule is associated with a frequency and a probability. Sets of predicates can attempt to identify an arbitrary number of agents. The relative location, i.e. the two-dimensional coordinates, of the node on the left-hand side of the rule is matched with its appearance on the right-hand side of the rule. If a node does not reappear on the right-hand side, it implies that its corresponding agent has been removed. If a node appears at a location that is unoccupied on the left-hand side, a new node is created. Figure 1 shows an example rule: It is applied with a probability of $p = 0.3$ at every fourth time step ($\Delta t = 4$). One (arbitrarily chosen) node that fulfills *predicateX* and *predicateY* is affected by *actionJ* and *actionK*. Also note that a new node is created and is initialized in this rule for which no reference had existed before. In case there are at least 6 nodes that fulfill *predicateZ*, they will all be removed.

3.2 Algorithm

A swarm graph grammar $SGG = (\mathcal{I}, \Xi, \mathcal{G}_{predicate}, \mathcal{G}_{action}, P)$ is a quintuple, where $\mathcal{I}$ describes a set of individuals relying on rules and properties as explained in the previous section. At the beginning of the simulation, a set Ξ of axioms, in the form of initialization algorithms, is executed by (1) selecting and expressing individuals from $\mathcal{I}$, and (2) by assigning initial states to the newly created individuals. For a homogeneous swarm of nest-constructing wasps[1], for instance, $\mathcal{I}$ only has to comprise a single agent description. Having created a sufficient number of wasp agents, the axioms would assign contextual information such as an initial location to the individuals. In the main loop of the swarm graph grammar algorithm (Algorithm 1) two graphs $G_{predicate} \in \mathcal{G}_{predicate}$ and $G_{action} \in \mathcal{G}_{action}$ are subsequently created that merge the triggered predicates

[1] See Section 4.2 for details.

and corresponding actions of the individuals' local rules. $\mathcal{G}_{predicate}$ represents the set of possible graphs of individuals interconnected through predicates. $\mathcal{G}_{action}$ hosts all possible action graphs. Chains of relations among sets of swarm individuals create semantic topologies for global graph structures that describe the situational context or activity in the SGG system. Executing the actions of G_{action} yields the next simulation state after a policy P is applied to resolve possibly arising computational conflicts[2]. Thus, the alternating update of the graph instances $G_{predicate}$ and G_{action} based on the swarm individuals' behaviours drives the SGG simulation (Figure 2).

Algorithm 1. Swarm Graph Grammar: Main Loop

Require: $G_{predicate}$, optional: P
Ensure: alternating computation of $G_{predicate}$ and G_{action}
  repeat
    compute predicative graph $G_{predicate}$
    compute action graph G_{action} based on $G_{predicate}$
    apply order policy P to G_{action}
    execute ordered actions of G_{action}
  until simulation is terminated

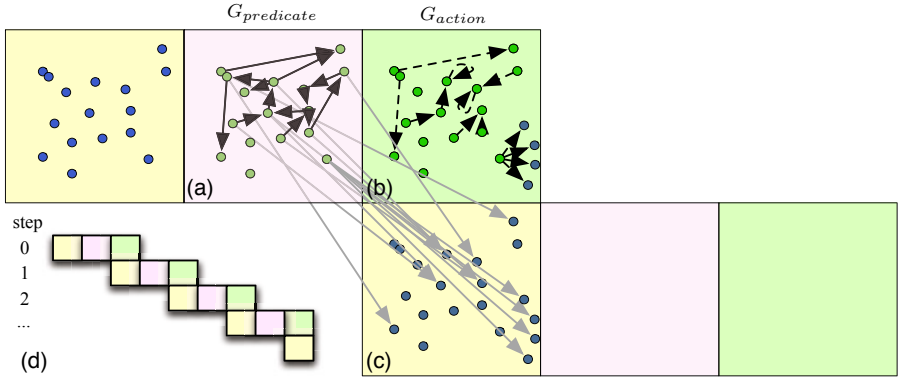


Fig. 2. Subsequent computation of (a) $G_{predicate}$ and (b) G_{action} yield (c) the next simulation state. The grey arrows from (a) to (c) relate nodes to their contextual impact. (d) The simulation process is shown as a computation pipeline.

4 Swarm Graph Grammars in Action

In this section we present three computational models realized with the SGG framework. We retrace (1) a simple boids simulation [30], (2) the stigmergic construction behaviour of the Chartergus wasp [35], and (3) cell proliferation induced by a set of growth factors.

[2] The implementation of an efficient conflict policy P is often difficult and its execution can be computationally expensive.

4.1 Boids

In order to specify a standard boid flocking simulation [30], we use a swarm graph grammar $SGG_{boid} = (\mathcal{I}_{boid}, \Xi_{boid}, \mathcal{G}^{boid}_{predicate}, \mathcal{G}^{boid}_{action}, P_{boid})$. The sole individual $i_{boid} \in \mathcal{I}_{boid}$ contains several weights for flocking urges, parameters to determine a field of perception, as well as boundaries for the maximal flight acceleration max_{accel} and velocity max_{vel}. Ξ_{boid} generates a homogeneous set of swarm individuals that are initialized with a random position $\overrightarrow{p}$ and velocity $\overrightarrow{v}$. As no interaction conflicts arise, the policy P is empty.

Boids rely on two behavioural rules shown in Figure 3. The *movement* rule continuously updates a swarm individual's position in accordance with its velocity. The *acceleration* rule, substitutes the predicate $sees(u, v)$ with the action $accelerate(u, v)$.

The predicate considers the reference node's location, orientation and perceptional field to select a set of interaction partners in accordance with their respective locations. The action also considers the difference between u's and v's states, including their locations and velocities, and accelerates u accordingly. For example, u accelerates towards v's location and it aligns its flight direction. In the example displayed in Figure 4, the boid agents form a cluster over time which is also reflected by increasingly connected interaction graphs.

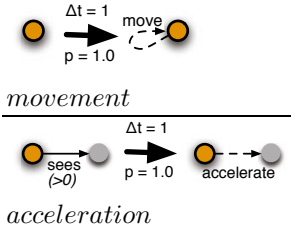


Fig. 3. Two rules to describe a boid agent's interaction behaviour

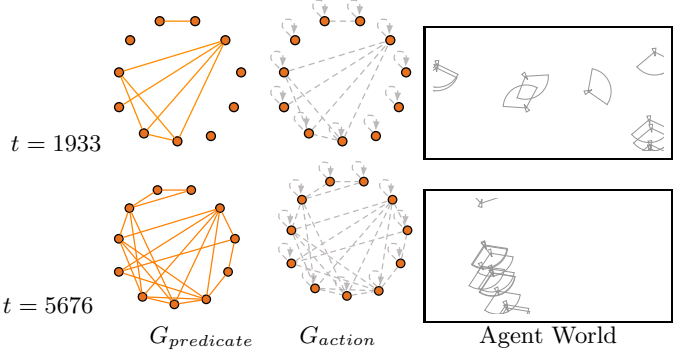
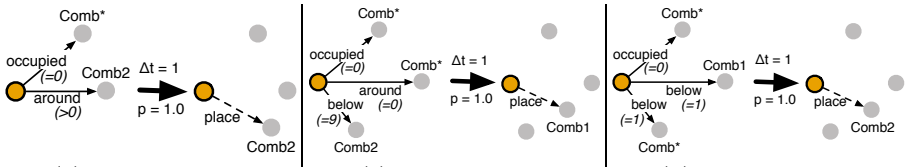


Fig. 4. Two sets of graphs $G_{predicate}$, G_{action} and a visualization of the agent space show a clustering process in a SGG-driven boid simulation. The boid renderings—triangles oriented towards their velocity with a conic field of perception—partially overlap due to their strong alignment urge.



(a) floor extension (c) floor template (d) floor initiation

Fig. 5. SGG rules that retrace the construction behaviour by the Chartergus wasp as described in [35]

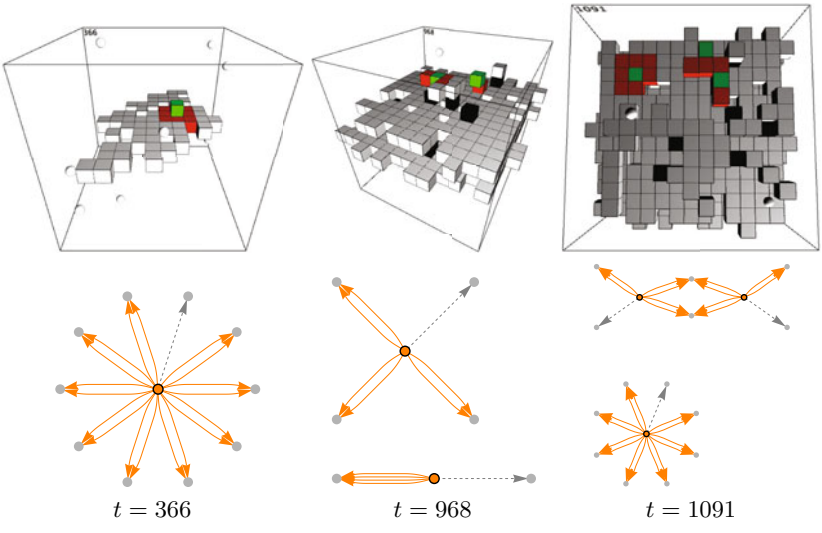


Fig. 6. Agent space and the corresponding interaction graphs of a wasp-inspired construction process (grey dashed arrows indicate actions, orange ones predicates). At $t = 366$ a floor template is constructed (rule (c) in Fig. 5). At $t = 968$ the construction of a new floor is started (rule (d) in Fig. 5). At $t = 1091$ two floor extensions are performed by different wasp agents triggered by the same subset of combs.

4.2 Stigmergic Construction

Theraulaz et al. have translated the nest construction processes of Chartergus wasps into individual behavioural rules [35]. The rules in Figure 5 closely retrace this behaviour[3]. The predicates *around*, *below* and *occupied* test the immediate surroundings of the wasp to trigger comb construction in the remaining rules. Hereby, previously deposited combs of two different types ($Comb1$, $Comb2$, or $Comb*$ for both) trigger the next *placement* actions. In addition, a *movement* rule as seen in Figure 3 moves an individual unconditionally to a random location in the simulation space. Figure 6 shows the development in agent space and

[3] The lattice-based matrix representation provided in [35] was translated into predicates that test the corresponding spatial relationships.

correlates the activating (red) and the constructed combs (green). The rule deployment is shown in a series of interaction graphs $G_{interaction} = G_{predicate} \cup G_{action}$.

4.3 Swarm Development

Signalling factors determine the rate of cell proliferation which influence specific morphological developments [26]. The rules in Figure 7 configure cells which *grow* until they reach maturity (predicates *not mature* and *mature*). Mature cells that are *close to* a *Growth Factor* increase their internal *mitogen* concentration which in turn instigates proliferation (modelled as *reset* of the acting cell and *initialization* of a second cell).

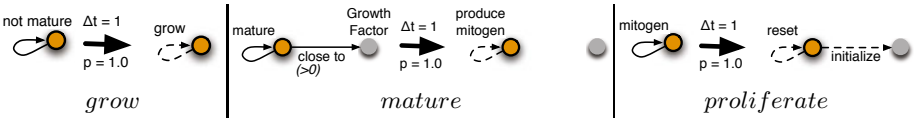


Fig. 7. Three rules to describe a simple developmental process model

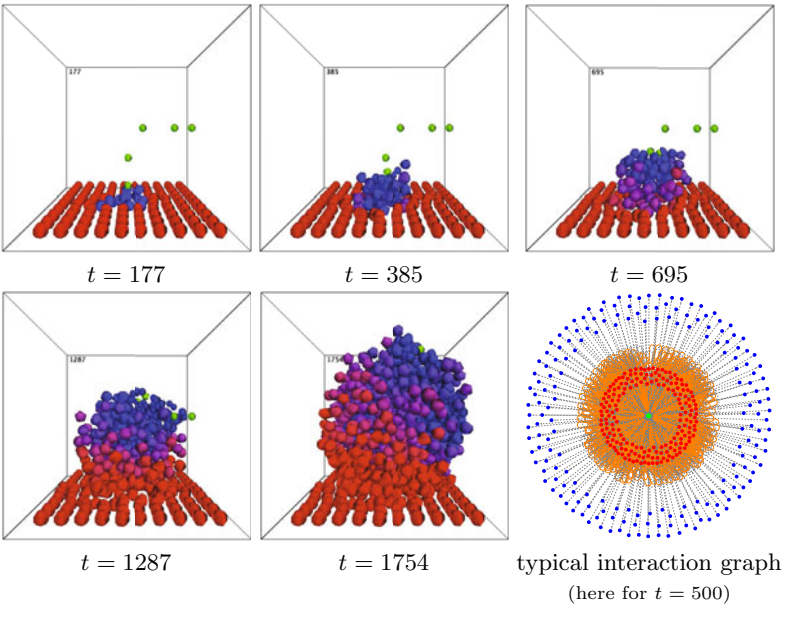


Fig. 8. The proliferation of mature cells (blue: not mature; red: mature) is dependent on the proximity to growth factors (green). At any time of the simulation, large numbers of agents are informed by growth factors leading to typically dense but homogeneous interaction graphs.

Figure 8 shows screenshots of the simulation. Tissue cells within the vicinity of a signalling molecule start proliferating. Collision resolution through an embedded physics engine allows the cells to assemble[4]. The emerging protuberance is slanted to the right in accordance to the initial distribution of signalling molecules. However, it is surprisingly symmetrical still, which could result from a lack of simulated cell polarization.

5 Summary and Future Work

Swarm graph grammars are a modelling and simulation framework that provides a universal graph-based representation for swarm-based developmental systems. Besides metabolic operations, i.e. the creation or removal of agents, the semantics of agent relations are not part of the framework. The agents' abilities have to be implemented in the form of predicates and actions. The agents' rule sets (behaviours) drive the simulation processes and they are immediately reflected in the interaction graph of a simulation. As examples, we used SGGs to simulate boid flocking, stigmergic wasp nest construction, and growth and proliferation in cellular morphological processes.

We are currently working on several aspects to improve and harness the utilization of swarm graph grammars. The application of the framework has led to many refinements in respect to the formalism and the algorithm. However, in order to render modelling with SGGs accessible, especially to non computer scientists, we need to collect feedback from interdisciplinary modellers about the shortcomings of the representation, e.g. regarding its visualization, terminology and logic. In this paper, we have touched upon matching local agent rules with a simulation's emerging interaction graphs. We deem this a very promising approach to analyze emergent phenomena in simulations on the one hand, and to create complex interaction processes with dynamic interaction topologies on the other hand. Accordingly, systematic investigations have to be started. We are also working on a slight modification of the SGG framework so that nodes can encapsulate children and thereby computational or spatial hierarchies can be built. This would allow for hierarchical modelling as in P systems [29].

Acknowledgements. Support for this research was provided by the Undergraduate Medical Eduction program of the University of Calgary. We would like to thank Jörg Denzinger for his invaluable advice on multi-agent systems and Heather Jamniczky for her feedback on biological developmental systems.

References

1. Allen, L.J.S.: An Introduction to Stochastic Processes with Applications to Biology. Pearson Education, Upper Saddle River (2003)
2. de Boer, M.J.M., de Does, M.: The relationship between cell division pattern and global shape of young fern gametophytes. I. A model study. Botanical Gazette 151(4), 423–434 (1990)

[4] In the given experiment we rely on the Bullet physics engine, http://bulletphysics.org

3. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, New York (1999)
4. Burleigh, I., Suen, G., Jacob, C.: Dna in action! a 3D swarm-based model of a gene regulatory system. In: ACAL 2003, First Australian Conference on Artificial Life, Canberra, Australia (2003)
5. Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton Studies in Complexity. Princeton University Press, Princeton (2003)
6. Culik, K., Lindenmayer, A.: Parallel graph generating and graph recurrence systems for multicellular development. International Journal of General Systems 3(1), 53–66 (1976)
7. Denzinger, J., Kordt, M.: Evolutionary on-line learning of cooperative behavior with situation-action-pairs. In: ICMAS, pp. 103–110. IEEE Computer Society, Los Alamitos (2000)
8. Denzinger, J., Winder, C.: Combining coaching and learning to create cooperative character behavior. In: CIG. IEEE, Los Alamitos (2005)
9. Ehrig, H., Kreowski, H.J., Montanari, U., Rosenberg, G. (eds.): Handbook of Graph Grammars and Computing by Fraph Transformation, Concurrency, Parallelism, and Distribution, vol. 3. World Scientific Publishing, Singapore (1999)
10. Giavitto, J.L., Godin, C., Michel, O., Prusinkiewicz, P.: Computational Models for Integrative and Developmental Biology. In: Modelling and Simulation of biological processes in the context of genomics, Hermes, pp. 12–17 (July 2002)
11. Giavitto, J.L., Michel, O.: Data structure as topological spaces. Unconventional Models of Computation, 137–150 (2002)
12. Giavitto, J.L., Michel, O.: Modeling the topological organization of cellular processes. Biosystems 70(2), 149–163 (2003)
13. Jacob, C., Burleigh, I.: Biomolecular swarms: An agent-based model of the lactose operon. Natural Computing 3(4), 361–376 (2004)
14. Jacob, C., Hushlak, G., Boyd, J., Nuytten, P., Sayles, M., Pilat, M.: Swarmart: Interactive art from swarm intelligence. Leonardo 40(3) (2007)
15. Jacob, C., Steil, S., Bergmann, K.: The swarming body: Simulating the decentralized defenses of immunity. In: Bersini, H., Carneiro, J. (eds.) ICARIS 2006. LNCS, vol. 4163, pp. 52–65. Springer, Heidelberg (2006)
16. Kauffman, S.: At Home in the Universe: The Search for the Laws of Self-Organization and Complexity. Oxford University Press, Oxford (1995)
17. Kniemeyer, O., Barczik, G., Hemmerling, R., Kurth, W.: Relational Growth Grammars – A Parallel Graph Transformation Approach with Applications in Biology and Architecture. In: Schürr, A., Nagl, M., Zündorf, A. (eds.) AGTIVE 2007. LNCS, vol. 5088, pp. 152–167. Springer, Heidelberg (2008)
18. Kniemeyer, O., Buck-Sorlin, G., Kurth, W.: Groimp as a platform for functional-structural modelling of plants. In: Vos, J., Marcelis, L.F.M., de Visser, P.H.B., Struik, P.C., Evers, J.B. (eds.) Functional-Structural Plant Modelling in Crop Production, pp. 43–52. Springer, Heidelberg (March 2006)
19. Kniemeyer, O., Buck-Sorlin, G.H., Kurth, W.: A graph grammar approach to artificial life. Artificial Life 10(4), 413–431 (2004)
20. Kumar, S., Bentley, P. (eds.): On Growth, Form and Computers. Elsevier Academic Press, London (2003)

21. Kurth, W., Buck-Sorlin, G., Kniemeyer, O.: Relationale wachstumsgrammatiken: Ein formalismus zur spezifikation multiskalierter Struktur-Funktions-Modelle von pflanzen. In: Modellierung pflanzlicher Systeme aus historischer und aktueller Sicht. Landwirtschaft, vol. 7, pp. 36–45. Landesamtes für Verbraucherschutz, Landwirtschaft und Flurneuordnung, Brandenburg (2006)
22. Lindenmayer, A.: Developmental systems without cellular interactions, their languages and grammars. Journal of Theoretical Biology 30(3), 455–484 (1971)
23. Lindenmayer, A.: An introduction to parallel map generating systems. Graph-Grammars and Their Application to Computer Science, 27–40 (1987)
24. von Mammen, S., Jacob, C.: The evolution of swarm grammars: Growing trees, crafting art and bottom-up design. IEEE Computational Intelligence Magazine 4(3), 10–19 (2009)
25. von Mammen, S., Jacob, C., Kókai, G.: Evolving swarms that build 3D structures. In: IEEE Congress on Evolutionary Computation, CEC 2005, pp. 1434–1441. IEEE Press, Edinburgh (2005)
26. Megason, S.G., McMahon, A.P.: A mitogen gradient of dorsal midline wnts organizes growth in the CNS. Development 129, 2087–2098 (2002)
27. Nagl, M.: On the relation between graph grammars and graph L-systems. Fundamentals of Computation Theory, 142–151 (1977)
28. von Neumann, J., Burks, A.W.: Theory of self-reproducing automata. University of Illinois Press, Urbana (1966)
29. Paun, G., Rozenberg, G.: A guide to membrane computing. Theoretical Computer Science 287(1), 73–100 (2002)
30. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. Computer Graphics 21(4), 25–34 (1987)
31. Salazar-Ciudad, I.: Tooth Morphogenesis in vivo, in vitro, and in silico. Current Topics in Developmental Biology 81, 341–371 (2008)
32. Sayama, H., Laramee, C.: Generative network automata: A generalized framework for modeling adaptive network dynamics using graph rewritings. Adaptive Networks, 311–332 (2009)
33. Schlick, T.: Molecular Modeling and Simulation: an interdisciplinary guide. Interdisciplinary Applied Mathematics, vol. 21. Springer, New York (2002)
34. Spicher, A., Michel, O., Giavitto, J.L.: A topological framework for the specification and the simulation of discrete dynamical systems. Cellular Automata, 238–247 (2004)
35. Theraulaz, G., Bonabeau, E.: Modelling the collective building of complex architectures in social insects with lattice swarms. Journal of Theoretical Biology 177(4), 381–400 (1995)
36. Tomita, K., Kurokawa, H., Murata, S.: Graph-rewriting automata as a natural extension of cellular automata. Adaptive Networks, 291–309 (2009)
37. Walker, D.C., Southgate, J.: The virtual cell–a candidate co-ordinator for 'middle-out' modelling of biological systems. Briefings in Bioinformatics 10(4), 450–461 (2009)
38. Wolfram, S.: A new kind of science. Wolfram Media Inc., Champaign (2002)
39. Wooldridge, M.: An Introduction to MultiAgent Systems. John Wiley and Sons, Chichester (2002)

A Modified Particle Swarm Optimization Algorithm for the Best Low Multilinear Rank Approximation of Higher-Order Tensors

Pierre B. Borckmans, Mariya Ishteva, and Pierre-Antoine Absil

Department of Mathematical Engineering,
Université catholique de Louvain, Louvain-la-Neuve, Belgium
http://www.inma.ucl.ac.be/~absil/

Abstract. The multilinear rank of a tensor is one of the possible generalizations for the concept of matrix rank. In this paper, we are interested in finding the best low multilinear rank approximation of a given tensor. This problem has been formulated as an optimization problem over the Grassmann manifold [14] and it has been shown that the objective function presents multiple minima [15]. In order to investigate the landscape of this cost function, we propose an adaptation of the Particle Swarm Optimization algorithm (PSO). The Guaranteed Convergence PSO, proposed by van den Bergh in [23], is modified, including a gradient component, so as to search for optimal solutions over the Grassmann manifold. The operations involved in the PSO algorithm are redefined using concepts of differential geometry. We present some preliminary numerical experiments and we discuss the ability of the proposed method to address the multimodal aspects of the studied problem.

Keywords: Multi-Linear Rank, Higher-Order Tensors, Particle Swarm Optimization, Grassmann Manifold, Global Optimization.

1 Introduction

Although the generalization of matrices to higher-order tensors is quite natural, the concept of matrix rank cannot be extended in a straightforward way. However, there are some propositions to define the rank for tensors, in an effort to maintain some properties found in the matrix theory. A particular choice, explained in section 2.1, is the multilinear rank, which is a direct generalization of the row and column ranks for matrices [12,13].

The problem we investigate in this paper is to find the best low multilinear rank approximation of a given tensor. This problem, presented in section 2.2, is a natural extension of the low rank matrix approximation problem. Amongst the many motivations behind these problems, one can cite data compression, dimensionality reduction and independent component analysis (ICA) in domains as various as statistics [20], signal processing [3] and image processing [24].

In the matrix case, it is known that the best approximation is given by the truncated singular value decomposition (SVD). The generalization of this concept for higher-order tensors is called the higher-order SVD (HOSVD) [6], or more generally Tucker decomposition [21,22]. However, taking truncated versions of these decompositions no longer leads to optimal approximations for

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 13–23, 2010.

tensors. That being said, the search for low rank tensor approximation can be formulated as an optimization problem on a particular search space, namely the Grassmann manifold, as explained in section 2.4.

This optimization problem has been studied before, notably in [7,18,17], and has been shown to present multiple local minima. In [15], the authors show that there can be many of these local minima but that the differences in terms of the cost function can be very small. What really differs are the subspaces spanned by the projection matrices corresponding to different solutions. These differences may have important consequences for applications and motivate the need to investigate the landscape of the objective function, without getting trapped in the first local minima. In order to perform this investigation, one needs to design global optimization methods on the Grassmann manifold. Some methods dealing with such search spaces are available in the literature. The global aspect of these methods rely either on a stochastic component or on a population framework. In [19] for example, the authors propose a stochastic gradient algorithm. In [9] Dreisigmeyer *et al.* define a canvas to develop direct-search optimization method on Riemannian manifolds, such as the Nelder-Mead algorithm and the lower-triangular mesh adaptive direct search (LTMADS) algorithm [4].

This paper presents preliminary results towards applying swarm intelligence techniques to efficiently explore the landscape of the objective function appearing in the low multilinear rank approximation problem. In this proof-of-concept paper, we concentrate on the most basic Particle Swarm Optimization (PSO) algorithms, extend them to tackle the tensor low rank approximation cost function defined on a Cartesian product of Grassmann manifolds, and report on numerical experiments that illustrate the applicability of the obtained algorithms. More specifically, we start from the Guaranteed Convergence PSO (GCPSO), as presented by van den Bergh *et al.* in [23], and modify it so as to search for optimal solutions over the Grassmann manifold. A gradient component is also included in the method to improve its convergence rate.

In future work, we will investigate methods that improve the exploration-exploitation balance, such as velocity-based reinitialization and niching techniques, and report on comparisons with other optimization methods.

The paper is organized as follows. Section 2 reviews the best low multilinear rank approximation of tensors, its formulation on the Grassmann manifold, and the issue of local minima. Section 3 presents how GCPSO is adapted to the Grassmann manifold, using concepts of differential geometry, and introduces a gradient component in the algorithm. Finally, section 4 presents numerical results and discusses the ability of the proposed method to deal with the multimodal aspects of the best low multilinear rank approximation problem.

2 Low Multilinear Rank Approximation Problem

2.1 Tensor Generalities

Before diving into the problem we want to address, we give here some general concepts and definitions about tensors that will be necessary in the following.

An *Nth-order real tensor* $\mathcal{A}$ is a N-way array of real numbers; a first-order tensor is thus a vector, a second-order tensor is a matrix, a third-order tensor is a 3D-array. In the following, we will focus on third-order tensors, for clarity, but extension to higher-order tensors is mainly technical. The *mode-n rank* of $\mathcal{A}$, $n = 1, \ldots, N$, denoted by $rank_n(\mathcal{A})$, is the number of linearly independent *mode-n* vectors, also called fibers, *i.e.* the vectors obtained by varying the n-th index of $\mathcal{A}$ while fixing the others. This concept is an extension of the row and column ranks of a matrix, with the important difference that the different mode-n ranks are generally not the same. The *multilinear rank* of an Nth-order tensor $\mathcal{A}$ is then the N-tuple of its *mode-n* ranks.

The *matrix representations* $A_{(n)}, n = 1, 2, 3$, of a third-order tensor $\mathcal{A}$, also called *flattenings* or *unfoldings* of $\mathcal{A}$, are obtained by concatenating the *mode-n* vectors of $\mathcal{A}$ in a specific order, given by:

$$\left(A_{(1)}\right)_{i_1,(i_2-1)I_3+i_3} = \left(A_{(2)}\right)_{i_2,(i_3-1)I_1+i_1} = \left(A_{(3)}\right)_{i_3,(i_1-1)I_2+i_2} = a_{i_1 i_2 i_3}. \quad (1)$$

Note that it results from this definition that $rank_n(\mathcal{A}) = rank(A_{(n)})$.

2.2 Problem Statement

Finding the best low multilinear rank of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ consists in finding another tensor $\hat{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ with mode-n ranks bounded by pre-specified R_1, R_2, R_3 respectively, with $R_i \leq I_i$, that is as close as possible to $\mathcal{A}$. This can be expressed by the following optimization problem:

$$\begin{aligned} &\min_{\hat{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}} \|\mathcal{A} - \hat{\mathcal{A}}\|^2 \\ \text{s.t.} \quad &rank_1(\hat{\mathcal{A}}) \leq R_1,\ rank_2(\hat{\mathcal{A}}) \leq R_2,\ rank_3(\hat{\mathcal{A}}) \leq R_3. \end{aligned} \quad (2)$$

It has been shown in [7,18,17] that this problem is equivalent to the problem of maximizing the function

$$\begin{aligned} &\bar{g} : St(R_1, I_1) \times St(R_2, I_2) \times St(R_3, I_3) \to \mathbb{R}, \\ &(U, V, W) \mapsto \|U^T A_{(1)}(V \otimes W)\|^2 = \|V^T A_{(2)}(W \otimes U)\|^2 \\ &\qquad\qquad = \|W^T A_{(3)}(U \otimes V)\|^2, \end{aligned} \quad (3)$$

where $St(p, n)$ stands for the set of column-wise orthonormal $(n \times p)$-matrices, *i.e.* the Stiefel manifold, and $\otimes$ is the Kronecker product.

This function $\bar{g}$ can be seen as a natural generalization of the matrix case where the best rank-R approximation $\hat{A}$ of a matrix $A \in \mathbb{R}^{I_1 \times I_2}$ is given by maximizing $\|U^T A V\|$, with $U \in \mathbb{R}^{I_1 \times R}$ and $V \in \mathbb{R}^{I_2 \times R}$ being column-wise orthogonal matrices (see [8]). In fact, in this case, the best solution is known to be given by the truncated SVD. For higher-order tensors, this notion can be extended to HOSVD [6,21,22], but it no longer leads to optimal solutions. Nevertheless, the truncated HOSVD is often used as a good starting point for optimization algorithms, even though it has been shown that this does not always imply convergence to the global optimum [15].

2.3 Local Minima

In [15], the authors show that the best low multilinear rank approximation problem (2) presents local non-global minima. This is a major difference with the matrix case where the problem is uni-modal. The authors considered tensors with low multilinear rank, perturbed by a small amount of additive noise. For such test tensors, the local algorithms proposed in [14] converge to a small amount of different minima. After increasing the noise level, the tensors become less structured and more local minima are found. This behavior is related to the distribution of the mode-n singular values (the singular values of the matrices $A_{(n)}$). When the noise level is low, there is a gap between the singular values, whereas when the noise level increases, the gap becomes small or non-existent. In this case, the best low multilinear rank approximation becomes a difficult problem since we are looking for a structure that is not present: there are many equally good, or equally bad, solutions.

Concerning the values of the cost function at the different local minima, they seem to be similar [15]. This means that in applications where the multilinear rank approximation is merely used as a compression tool for memory savings, using a local optimizer such as the ones proposed in [14] is efficient and reliable.

On the other hand, the subspaces spanned by two matrices U_1 and U_2 corresponding to two different local optima in (3) are very different and the same holds for V and W [15]. In applications where these subspaces are important, such as dimensionality reduction, this observation may be important to consider. A last remark made in [15] is that the global optimum is not necessarily the desired one. The authors show that when a tensor with low multilinear rank is affected by noise, the subspaces corresponding to the global optimum are not always the closest to the subspaces corresponding to the original noise-free tensor, especially for high noise levels.

All of these remarks motivate the need to develop methods that are able to investigate the landscape of the objective function (3) in a global perspective. In [14], the author suggests that local algorithms can be run several times, from different initial conditions, in order to discover the different local optima. In this paper, we propose a more sophisticated, population-based method, in the perspective that one run of such a global method with a given number of individuals would discover the different optima more efficiently than the same amount of runs of a local optimizer, thanks to collaboration between individuals.

2.4 The Grassmann Manifold $G(p, n)$

The cost function $\bar{g}$ defined in (3) has an important invariance property. For any set of orthogonal matrices $Q^{(i)} \in O_{R_i}, i = 1, 2, 3$, one can show that:

$$\bar{g}(U, V, W) = \bar{g}(UQ^{(1)}, VQ^{(2)}, WQ^{(3)}). \tag{4}$$

This means that we are not looking for the exact elements of U, V and W, but for the subspaces that their columns span. Consequently, the optimal solutions of $\bar{g}$ are not isolated since for every triplet (U, V, W), there is a whole equivalence

class of triplets leading to the same cost. For PSO methods, this is a source of difficulty since particles may converge to minima that are structurally identical (they yield the same cost) while being far apart from each other in the (U, V, W) space. One way of getting rid of this redundancy is to work conceptually on a smaller-dimensional search space by making use of the Grassmann manifold.

The Grassmann manifold $G(p, n)$ is the set of all p-planes in $\mathbb{R}^n$. It can be thought of as the quotient manifold $M = St(p, n)/O_p$ (see *e.g.* [10]). A point in $G(p, n)$ thus represents a whole equivalence class of matrices spanning the same p-dimensional subspace. In practice, we will represent a point in $G(p, n)$ by choosing an arbitrary member in $St(p, n)$ of its equivalence class.

3 Particle Swarm Optimization

In this section, we propose to develop an optimization algorithm based on PSO for the best low multilinear rank approximation problem. In order to do so, we need to adapt PSO to the cost function (3), defined on a Cartesian product of three Grassmann manifolds. This section recalls the basic PSO and GCPSO algorithm formulations, then presents the adaptation of their different components to the Grassmann manifold. Finally the algorithm for the Cartesian product is detailed.

3.1 PSO in $\mathbb{R}^n$

First introduced in 1995 by Eberhart and Kennedy [16], PSO is a stochastic population-based algorithm. Particles are points evolving in the search space, following simple rules, mimicking the behaviour of social groups. Points are initialized randomly in $\mathbb{R}^n$ and the driving force of the optimization process is given by the following update equations (iterated over k), for each particle (indexed by i):

$$v_i(k+1) = \underbrace{w(k)v_i(k)}_{inertia} + \underbrace{c\alpha_i(k)\,(y_i(k) - x_i(k))}_{nostalgia} + \underbrace{s\beta_i(k)\,(\hat{y}(k) - x_i(k))}_{social} \quad (5)$$

$$x_i(k+1) = x_i(k) + v_i(k+1)\,, \quad (6)$$

where x denotes position, v denotes velocity, y is the personal best position so far, $\hat{y}$ is the global best position discovered by the swarm so far; w is inertia coefficient (usually dynamic), c and s are adjustable coefficients, and α and β are random components. As can be seen, the behaviour of each particle is dictated by velocity increments composed of three simple components: inertia, cognitition (nostalgic behaviour) and social behaviour. At the end of each iteration, y_i and $\hat{y}$ are updated.

3.2 GCPSO and Gradient

The basic PSO algorithm introduced in the previous section does not present any guarantee about the convergence to a minimum, not even a local one. Actually,

the only thing that is guaranteed is that all the particles converge to a point on the line joining their best position and the global best position. In order to ensure convergence to a stationary point, van den Bergh *et al.* proposed in [23] a slight variation of the original PSO, called Guaranteed Convergence PSO, avoiding stagnation of the particles. In this model, all the particles follow the velocity update equation (5) except for the best particle at current iteration k, denoted by the index $i = \tau$, which follows the velocity equation

$$v_\tau(k+1) = \underbrace{\hat{y}(k) - x_\tau(k)}_{reset} + \underbrace{w(k)v_\tau(k)}_{inertia} + \underbrace{\rho(k)(1-2\gamma(k))}_{sampling}, \tag{7}$$

where γ is a random vector uniformly distributed over $[0, 1]$. This last equation forces the best particle to randomly reset its position in a ball around the current best position. The radius of this ball, $\rho(k)$, is increased when a better position is sampled and it is reduced when the sampling fails, therefore ensuring convergence to a local minimum. For more in-depth information about PSO, GCPSO and other variants, the reader is referred to [11].

When the gradient of the objective function is available, the sampling term in (7) can be advantageously replaced by a term involving the gradient:

$$v_\tau(k+1) = \underbrace{\hat{y}(k) - x_\tau(k)}_{reset} + \underbrace{w(k)v_\tau(k)}_{inertia} - \underbrace{\delta(k)\nabla f(x_\tau(k))}_{descent}, \tag{8}$$

where $\delta(k)$ controls how much use of the gradient is made over time. The idea there is that the whole swarm is moving through the search space using mutual interactions, while the best individual refines its position in the best way that is locally possible, *i.e.* along its steepest descent direction. This strategy has the advantage of preserving the exploration quality of PSO while improving the exploitation of potential optima that are found along the way.

3.3 Adapting GCPSO to the Grassmann Manifold $G(p, n)$

In its original form, PSO is described for particles in $\mathbb{R}^n$: $x \in \mathbb{R}^n$, $y \in \mathbb{R}^n$ and the operations involved in the update equations $(+,-,.)$ are the usual vectorial addition, subtraction and scaling. In order to adapt standard PSO to the Grassmannian search space, one must redefine x, v and the operations mentioned above.

A point x on the Grassmann manifold $G(p, n)$ is represented by a matrix $X \in St(p, n)$ and likewise for y and $\hat{y}$. The velocity v is now an element of $T_xG(p, n)$, the tangent space to $G(p, n)$ centered at x, represented by a matrix in $\mathcal{H}_{\mathbf{x}} := \left\{\dot{X} \in \mathbb{R}^{n\times p} : X^T\dot{X} = 0\right\}$; see [2, §3.6.2] for details. In the following, we will abuse notation and write $X \in G(p, n)$ for an element of $St(p, n)$ meant to represent an element of $G(p, n)$, and $\dot{X} \in T_XG(p, n)$ for an element of $\mathcal{H}_{\mathbf{x}}$ meant to represent an element of $T_XG(p, n)$.

In order to rewrite the velocity and position update equations (5)–(6), one needs to be able to compute a geodesic starting at a given point with a given

initial velocity, and to compute the initial velocity of a geodesic linking two points. The exponential map and the logarithmic map are the respective tools to address these questions (see *e.g.* [1]). We consider the geodesics induced by the (essentially) unique rotation-invariant metric on $G(p,n)$.

The geodesic starting at a point $X \in G(p,n)$ with velocity $\dot{X} \in T_X G(p,n)$ is given by (see [1, §3.7]):

$$\gamma(t) = \mathrm{Exp}_X(t\dot{X}) := XV\cos(t\Theta) + U\sin(t\Theta)\ , \tag{9}$$

where $\dot{X} = U\Theta V^T$ is a thin SVD.

The initial velocity $\dot{X} \in T_X G(p,n)$ of the geodesic linking two points X and Y on $G(p,n)$ is given by (see [1, §3.8]):

$$\dot{X} = \mathrm{Log}_X(Y) := U\mathrm{atan}(\Sigma)V^T\ , \tag{10}$$

where U, V and Σ are given by the thin SVD of $\Pi_X^{\perp} Y (X^T Y)^{-1}$ (where $\Pi_X^{\perp} := I - XX^T$ is the orthogonal projection onto $\mathcal{H}_{\mathbf{x}}$):

$$U\Sigma V^T = \Pi_X^{\perp} Y \left(X^T Y\right)^{-1}\ . \tag{11}$$

Using relations (9) and (10), the velocity and position update equations (5)–(6) can now be rewritten as follows:

$$\begin{aligned}\dot{X}_i(k+1) = w(k)\tfrac{\mathrm{d}}{\mathrm{dt}}\,\mathrm{Exp}_{X_i(k-1)}\left(t\dot{X}_i(k)\right)\Big|_{t=1} &+\ c\alpha_i(k)\mathrm{Log}_{X_i(k)}Y_i(k)\\ &+\ s\beta_i(k)\mathrm{Log}_{X_i(k)}\hat{Y}(k)\end{aligned} \tag{12}$$

$$X_i(k+1) = \mathrm{Exp}_{X_i(k)}\dot{X}_i(k+1). \tag{13}$$

Equation (7) for GCPSO involves three terms. The first two terms are similar to the ones in equation (5) and can therefore be adapted in the same way. To adapt the last term (sampling), one needs to be able to generate a displacement in $T_X G(p,n)$, the tangent space at X, so as to generate a random matrix $Y \in G(p,n)$ in a ball of radius ρ around $X \in G(p,n)$. This displacement Z can be computed as $Z = U\Sigma V^T$ where U, Σ and V are chosen as follows: draw $R \in \mathbb{R}^{n\times p}$ and $W \in \mathbb{R}^{p\times p}$ from the normal distribution, $(s_1,\ldots,s_p)$ from the uniform distribution on $[0,1]$ and compute:

$$\begin{array}{c} U = \mathrm{qf}(Q) \\ Q = \Pi_X^{\perp}(R) \end{array} \qquad \Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_p \end{bmatrix} \qquad V = \mathrm{qf}(W) \tag{14}$$
$$\sigma_i = \rho(k)(1-2s_i)$$

where qf returns the Q-factor of the QR decomposition of its matrix argument.

Equation (7) can thus finally be rewritten as follows:

$$\dot{X}_\tau(k+1) = \mathrm{Log}_{X_\tau(k)}\hat{Y}(k)\ +\ w(k)\frac{\mathrm{d}}{\mathrm{dt}}\,\mathrm{Exp}_{X_\tau(k-1)}\left(tV_\tau(k)\right)\Big|_{t=1}\ +\ U\Sigma_{\rho(k)}V^T. \tag{15}$$

Finally, equation (8) can be rewritten in the following way:

$$\dot{X}_\tau(k+1) = \mathrm{Log}_{X_\tau(k)} \hat{Y}(k) + w(k) \frac{\mathrm{d}}{\mathrm{d}t} \mathrm{Exp}_{X_\tau(k-1)} \left(tV_\tau(k)\right)\Big|_{t=1} - \delta(k) \nabla f(X_\tau(k)). \tag{16}$$

3.4 PSO in $G(R_1, I_1) \times G(R_2, I_2) \times G(R_3, I_3)$

The previous section showed how PSO and GCPSO can be adapted to the Grassmann manifold. For the purpose of optimizing $\bar{g}$ (3), the search space is actually a Cross product of three Grassmann manifolds, $G(R_1, I_1) \times G(R_2, I_2) \times G(R_3, I_3)$ and we endow this search space with the standard product metric. The particles are thus elements described as follows:

$$\{(U, V, W) : U \in G(R_1, I_1), V \in G(R_2, I_2), W \in G(R_3, I_3)\}.$$

The formulas for the exponential and the logarithm generalize component-wise as follows:

$$\mathrm{Exp}_{(U,V,W)} \left(\dot{U}, \dot{V}, \dot{W}\right) = \left(\mathrm{Exp}_U \dot{U}, \mathrm{Exp}_V \dot{V}, \mathrm{Exp}_W \dot{W}\right), \tag{17}$$

$$\mathrm{Log}_{(U_a,V_a,W_a)} (U_b, V_b, W_b) = \left(\mathrm{Log}_{U_a} U_b, \mathrm{Log}_{V_a} V_b, \mathrm{Log}_{W_a} W_b\right). \tag{18}$$

Equations (12)-(16) can now be adapted to the search space using (17)-(18).

Using equations (12)-(18), we finally propose the following algorithm:

Data: Tensor $\mathcal{A}$, Multilinear rank (R_1, R_2, R_3)
Draw initial points $(U_i(0), V_i(0), W_i(0))$, $\forall i$, from the normal distribution and taking the Q-factor;
Set null initial velocities $(\dot{U}_i(0), \dot{V}_i(0), \dot{W}_i(0))$;
while $k < max\_iter$ **do**
  Update personal bests $\left(Y_i^U(k), Y_i^V(k), Y_i^W(k)\right)$, $\forall i$;
  Update global best $\left(\hat{Y}^U(k), \hat{Y}^V(k), \hat{Y}^W(k)\right)$ and τ;
  $\forall i \neq \tau$, compute $\left(\dot{U}_i(k), \dot{V}_i(k), \dot{W}_i(k)\right)$ using (12);
  if $k \leq iter\_GCPSO$ **then**
    Compute $\left(\dot{U}_\tau(k), \dot{V}_\tau(k), \dot{W}_\tau(k)\right)$ using (15) *(sampling)*;
  else
    Compute $\left(\dot{U}_\tau(k), \dot{V}_\tau(k), \dot{W}_\tau(k)\right)$ using (16) *(gradient)*;
  end
  Compute $(U_i(k), V_i(k), W_i(k))$ using (13);
  Update $\rho(k)$ or $\delta(k)$;
end
return $\bar{g}\left(U_\tau(k), V_\tau(k), W_\tau(k)\right)$

Algorithm 1. GCPSO with gradient adapted to the best low multilinear rank tensor approximation problem

4 Numerical Results

In this section, we present some preliminary numerical experiments that illustrate the ability of the proposed algorithm to find the global optimum of the cost function $\bar{g}$. We build a set of test tensors as follows:

- we choose dimensions (I_1, I_2, I_3), the multilinear rank $R = (R_1, R_2, R_3)$ and noise levels σ_i
- we build a set of tensors $\mathcal{A}_i$ with multilinear rank R, perturbed with additive noise controlled by σ_i, using

$$\tilde{\mathcal{A}}_i = \mathcal{T}_i/\|\mathcal{T}_i\| + \sigma_i * \mathcal{E}_i/\|\mathcal{E}_i\|, \quad \mathcal{A}_i = \tilde{\mathcal{A}}_i/\|\tilde{\mathcal{A}}_i\|, \tag{19}$$

  where $\mathcal{T}_i$ is obtained as a product of a (R_1, R_2, R_3)-tensor $\mathcal{B}_i$ and three column-wise orthonormal matrices with matching dimensions (to reach an (I_1, I_2, I_3)-tensor). The elements of $\mathcal{B}_i$, $\mathcal{E}_i$ and the three matrices are drawn from a normal distribution with zero mean and unit standard deviation. The Q-factor of the QR decomposition is then taken for the matrices.

For each tensor of this set, we perform 100 runs of a local optimizer, namely the higher order orthogonal iteration (HOOI) [7,17], with randomly-selected initial conditions, in order to get a putative global optimum (U_*, V_*, W_*). Then, we run the proposed algorithm 1, and we declare success if the best objective value $\bar{g}_\star$ found by the algorithm falls within $\epsilon = 10^{-8}$ of $\bar{g}(U_*, V_*, W_*)$.

The parameters for PSO were chosen as follows:

- $population\_size = 10$, $c = 0.5$, $s = 0.5$, $max\_iter = 1000$, $iter\_GCPSO = 400$;
- $w(k)$ linearly decreasing from 0.9 to 0.4;
- $\rho(k)$ doubling when 3 consecutive successes, halving when 5 consecutive failures;
- $\delta(k)$ linearly decreasing from 1 to 0.

The following table shows the rate of success (a rate of 1 means that success was always observed) for the cases $(I_1, I_2, I_3) = (10, 10, 10)$, $(R_1, R_2, R_3) = (2, 2, 2)$ and $(I_1, I_2, I_3) = (10, 12, 15)$, $(R_1, R_2, R_3) = (2, 3, 4)$, evaluated over 10 runs of algorithm 1.

	σ	1	2	3	5	10
case 1	success rate	1	1	.9	.4	0
case 2	success rate	1	1	.8	.3	.1

We see that algorithm 1 is very successful when the noise level is low, even though local non-global optima are already present in the objective function. We also see that when the noise level becomes very high, the landscape of the objective function becomes so intricate that the performance of the proposed algorithm degrades.

5 Conclusion

We have introduced an adaptation of the GCPSO algorithm, including a gradient component, for the best low multilinear rank approximation problem. The proposed algorithm shows capacities to discover global optima, often without getting trapped in suboptimal solutions.

In future work, we will explore the niching capabilities of the PSO algorithm, as presented *e.g.* in [25,5]. These variations improve the ability of PSO to discover different local minima and could be used in this application to investigate the landscape of the objective function in a more informative way.

Further investigation of the problem might also involve other optimization techniques such as evolutionary algorithms and memetic algorithms. A comparison with such existing population-based algorithms might give more insight about the difficulty of the considered problem.

Acknowledgments. This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with its authors. This work is supported by a FRIA (Fonds pour la formation à la Recherche dans l'Industrie et dans l'Agriculture) fellowship and by "Communauté française de Belgique - Actions de Recherche Concertées".

References

1. Absil, P.A., Mahony, R., Sepulchre, R.: Riemannian geometry of Grassmann manifolds with a view on algorithmic computation. Acta Appl. Math. 80(2), 199–220 (2004)
2. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press, Princeton (2008)
3. Acar, E., Bingol, C.A., Bingol, H., Bro, R., Yener, B.: Multiway analysis of epilepsy tensors. In: ISMB 2007 Conference Proceedings, Bioinformatics, vol. 23(13), pp. i10–i18 (2007)
4. Audet, C., Dennis Jr., J.E.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. on Optimization 17(1), 188–217 (2006)
5. Brits, R., Engelbrecht, A., van den Bergh, F.: A niching particle swarm optimizer. In: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), vol. 2, pp. 692–696 (2002)
6. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. 21(4), 1253–1278 (2000)
7. De Lathauwer, L., De Moor, B., Vandewalle, J.: On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors. SIAM J. Matrix Anal. Appl. 21(4), 1324–1342 (2000)
8. De Lathauwer, L., Vandewalle, J.: Dimensionality reduction in higher-order signal processing and rank-$(R_1, R_2, \ldots, R_N)$ reduction in multilinear algebra. Linear Algebra Appl. 391, 31–55 (November 2004), Special Issue on Linear Algebra in Signal and Image Processing

9. Dreisigmeyer, D.W.: Direct search algorithms over Riemannian manifolds (December 2006) (optimization Online 2007-08-1742)
10. Edelman, A., Arias, T.A., Smith, S.T.: The geometry of algorithms with orthogonality constraints. SIAM J. Matrix Anal. Appl. 20(2), 303–353 (1998)
11. Engelbrecht, A.P.: Fundamentals of Computational Swarm Intelligence. John Wiley & Sons, Chichester (2006)
12. Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products. Journal of Mathematical Physics 6(1), 164–189 (1927)
13. Hitchcock, F.L.: Multiple invariants and generalized rank of a p-way matrix or tensor. Journal of Mathematical Physics 7(1), 39–79 (1927)
14. Ishteva, M.: Numerical methods for the best low multilinear rank approximation of higher-order tensors. Ph.D. thesis, Department of Electrical Engineering, Katholieke Universiteit Leuven (December 2009)
15. Ishteva, M., Absil, P.-A., Van Huffel, S., De Lathauwer, L.: Tucker compression and local optima. Chemometr. Intell. Lab. Syst. (2010), doi:10.1016/j.chemolab.2010.06.006
16. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995), http://dx.doi.org/10.1109/ICNN.1995.488968
17. Kroonenberg, P.M.: Applied Multiway Data Analysis. Wiley, Chichester (2008)
18. Kroonenberg, P.M., de Leeuw, J.: Principal component analysis of three-mode data by means of alternating least squares algorithms. Psychometrika 45(1), 69–97 (1980)
19. Liu, X., Srivastava, A., Gallivan, K.: Optimal linear representations of images for object recognition. IEEE Pattern Anal. and Mach. Intell. 26(5), 662–666 (2004), http://dx.doi.org/10.1109/TPAMI.2004.1273986
20. McCullagh, P.: Tensor Methods in Statistics. Chapman and Hall, London (1987)
21. Tucker, L.R.: The extension of factor analysis to three-dimensional matrices. In: Gulliksen, H., Frederiksen, N. (eds.) Contributions to mathematical psychology, pp. 109–127. Holt, Rinehart & Winston (1964)
22. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika 31, 279–311 (1966)
23. van den Bergh, F., Engelbrecht, A.P.: A new locally convergent particle swarm optimiser. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. pp. 96–101 (2002)
24. Vasilescu, M.A.O., Terzopoulos, D.: Multilinear subspace analysis for image ensembles. In: Proc. Computer Vision and Pattern Recognition Conf. (CVPR 2003), Madison, WI, vol. 2, pp. 93–99 (2003)
25. Zhang, J., Zhang, J.R., Li, K.: A sequential niching technique for particle swarm optimization. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005. LNCS, vol. 3644, pp. 390–399. Springer, Heidelberg (2005)

A Robotic Validation of the Attractive Field Model: An Inter-disciplinary Model of Self-regulatory Social Systems

Md. Omar Faruque Sarker and Torbjørn S. Dahl

Robotic Intelligence Lab, University of Wales, Newport, UK
{Mdomarfaruque.Sarker,Torbjorn.Dahl}@newport.ac.uk

Abstract. Division of labour in multi-robot systems or multi-robot task allocation (MRTA) is a challenging research issue. We propose to solve this MRTA problem using a set of previously published generic rules for division of labour derived from the observation of ant, human and robotic social systems. These bottom-up rules describe the phenomenon of self-regulated division of labour in terms of attractive fields between robots and tasks. The concrete form of these rules, the *attractive filed model*, avoids the strong dependence on local interactions found in many existing approaches to MRTA. We present experimental results that constitute a first validation of attractive filed model as a mechanism for MRTA and as a multi-disciplinary model of self-organisation in social systems. Our experiments used 16 e-puck robots in a 2m x 2m area.

1 Introduction

Scientific studies show that a large number of animal as well as human social systems grow, evolve and generally continue functioning well by the virtue of their individual self-regulatory mechanism of division of labour (DOL) [2]. This has been accomplished without any central authority or any explicit planning and coordinating element. Indirect communication such as stigmergy is instead used to exchange information among individuals. In robotics, *multi-robot task allocation* (MRTA) is a common research challenge [4]. It is generally identified as the problem of assigning tasks to appropriate robots at appropriate times considering the potential changes in the environment and/or the performance of the robots. MRTA is an optimal assignment problem that has been shown to be NP-hard, so optimal solutions can not be expected for large problems [8]. In addition to the inherent complexity of MRTA, the problem is also commonly restricted to avoid central planners or coordinators for task assignments. The robots are also commonly limited to local sensing, communication and interaction [5] where no single robot has complete knowledge of the past, present or future actions of other robots or a complete view of the world state.

Traditionally MRTA solutions are broadly divided into two major categories: 1) Predefined and 2) Bio-inspired self-organized task-allocation [9]. Early research on predefined task-allocation was dominated by intentional coordination,

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 24–35, 2010.

use of dynamic role assignment [8] and market-based bidding approach [3]. Under these approaches, robots use direct task-allocation method to communicate and to negotiate tasks. These approaches are intuitive, comparatively straightforward to design and implement and can be analysed formally. However, these approaches typically works well only when the number of robots are small (≤ 10).

The self-organized task-allocation, on the other hand, relies on the emergence of group behaviours, e.g., emergent cooperation [5], adaptive mechanisms such as *adaptation rules* [6]. These approaches typically handle systems with local sensing, local interactions and typically little or no explicit communications or negotiations among robots. self-organized systems are more scalable and robust due to their inherent parallelism and redundancy. However, in these systems, solutions are unintuitive and thus difficult to design, analyse formally and implement practically [4,5]. The solutions found by these systems are typically sub-optimal and, as emergence is a result of interactions among robots and their environment, it is also difficult to predict exact behaviours of robots and overall system performance.

The current challenges in self-organized task allocation approaches have motivated us to look for a suitable alternative. In nature we find that task allocation in animal or insect societies can be governed by non-centralized rules and that they are self-regulating and self-stabilizing [2]. As a part of a collaborative project, we have studied the behaviours of ants, humans and robots and, from a set of generic rules of self-regulation, we have developed the attractive field model (AFM). This has become a common formal model of division of labour in social systems [1]. In this paper, we present an application of AFM in a robotic system. Section 2 presents AFM with its generic and robotic interpretations. It also presents an application of AFM in a manufacturing shop-floor scenario. Section 3 introduces our implementation of MRTA. Section 4 presents the design of our experiments including specific parameters and observables. Section 5 discusses our experimental results and section 6 draws conclusions.

2 The Attractive Field Model

2.1 Generic Interpretation

AFM provides an abstract framework for self-regulatory DOL in social systems [1]. In order to achieve self-regulation, it proposes four generic rules: continuous flow of information, concurrency, learning and forgetting, all of them will be explained later. In terms of networks, the model is a bipartite network, meaning that there are two different types of nodes. One set of nodes describes the sources of the attractive fields and the other set describes the agents. Links only exist between different types of nodes and they encode the flow of information so that, even if there is no direct link between two agents, their interaction is taken into account in the information flow. The strength of the field depends on the distance between the task and the agent. This relationship is represented using weighted links. In addition, there is a permanent field that represents the no-task option.

The model can be mapped to a network as shown in Fig. 1. The correspondence is given below:

1. Source nodes (o) are tasks that can be divided between a number of agents.
2. Agent nodes (x) are robots.
3. The attractive fields correspond to stimuli to perform a task, and these are encoded in a system-wide *continuous flow of information*. These are given by the black solid lines.
4. When an agent performs a task, the link is of a different sort, and this is denoted in the figure by a dashed line. Agents linked to a source by a red line are the robots currently doing that task.
5. The field of ignoring the information (w) corresponds to the stimulus to random walk, i.e. the no-task option, and this is denoted by the dotted lines in the graph.
6. Each of the links is weighted. The value of this weight describes the strength of the stimulus that the agent experiences. In a spatial representation of the model, it is easy to see that the strength of the field depends on the physical distance of the agent to the source. In addition, the strength can be increased through sensitisation of the agent via experience or *learning*. Similarly, when the task is not done for some time, this strength can be gradually attenuated, which can be treated as *forgetting*. This distance is not depicted in the network, it is represented through the weights of the links . In the figure of the network, the nodes have an arbitrary place. Note that even though the distance is physical in this case, the distance in the model applied to other systems, needs not to be physical, it can represent the accessibility to the information, the time the information takes to reach the receiver, etc.

In summary, looking at the network, we see that each of the agents is connected to each of the fields. This means that even if an agent is currently involved in a task, the probability that it stops doing it in order to pursue a different task, or to random walk, is always non-zero. The weighted links express the probability of an agent to be attracted to each of the fields, considered as *concurrency*.

2.2 Robotic Interpretation

Now let us interpret this model within the context of our MRS. Let us consider a manufacturing shop floor scenario where N number of mobile robots are required to attend to M number of shop tasks spread over a fixed area A. Let these tasks be represented by a set of small rectangular boxes resembling to manufacturing machines. Let R be the set of robots $r_1, r_2, ..., r_n$ and J the set of tasks $t_1, t_2, ..., t_m$. Each task t_j has an associated task-urgency ϕ_j indicating its relative importance over time. If a robot attends a task t_j in the x^{th} time-step, the value of ϕ_j will decrease by an amount $\Delta\Phi_{INC}$ in the $(x+1)^{th}$ time-step. On the other hand, if a task has not been served by any robot in the x^{th} time-step, ϕ_j will increase by another amount, $\Delta\Phi_{DEC}$, in $(x+1)^{th}$ time-step. In order to complete a task t_1, a robot r_i needs to be within a fixed boundary D_{j1} of t_1.

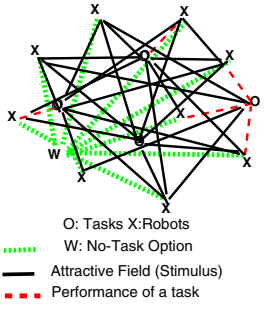


Fig. 1. Attractive Filed Model (AFM)

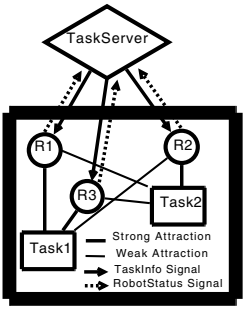


Fig. 2. A centralized communication scheme

If a robot completes a task t_j it gets sensitised to it and this will increase the robot's likelihood of selecting that task in the future. We call this variable the affinity of a robot r_i to task t_j its *sensitization* k_j^i. If a robot does not do a task t_j for some time, it forgets about t_j and k_j is decreased.

According to AFM, all robots will establish attractive fields to all tasks due to the presence of a system-wide continuous flow of information. The strength of these attractive fields will vary according to the distances between robots and tasks, task-urgencies and corresponding sensitizations of robots. This is encoded in Eq. 1.

$$S_j^i = tanh\{\frac{k_j^i}{d_{ij} + \delta}\phi_j\} \qquad (1) \qquad\qquad P_j^i = \frac{S_j^i}{\sum_j S_j^i} \qquad (2)$$

Eq. 1 states that the stimuli of a robot r_i to a particular task t_j, S_j^i, depends on r_i's spatial distance to t_j (d_{ij}), level of sensitization to j (k_j^i), and perceived urgency of that task (ϕ_j). We use a very small constant value δ to avoid division by zero for the special case when a robot has reached a task. Since S_j^i is a probability function, it is chosen as a *tanh* in order to keep the values between 0 and 1. The probability of selecting each task has been determined by a probabilistic method outlined in Eq. 2. AFM suggests that concurrency of a self-regulatory system can be maintained by specifying at least two task options: doing a task and not doing a task. In robots, the latter can be be treated as random walking. So in any time-step a robot will choose from M+1 tasks. Let T_a be the allocated time to accomplish a task. If a robot can enter inside the task boundary within T_a time it waits there until T_a elapsed. Otherwise it will select a different task.

2.3 A Manufacturing Shop-Floor Interpretation

By extending the robotic interpretation of AFM, we can present a manufacturing shop-floor scenario where each task represents a manufacturing machine. These machines are capable of producing goods from raw materials, but they also require constant maintenance works for stable operations. Let W_j be a finite

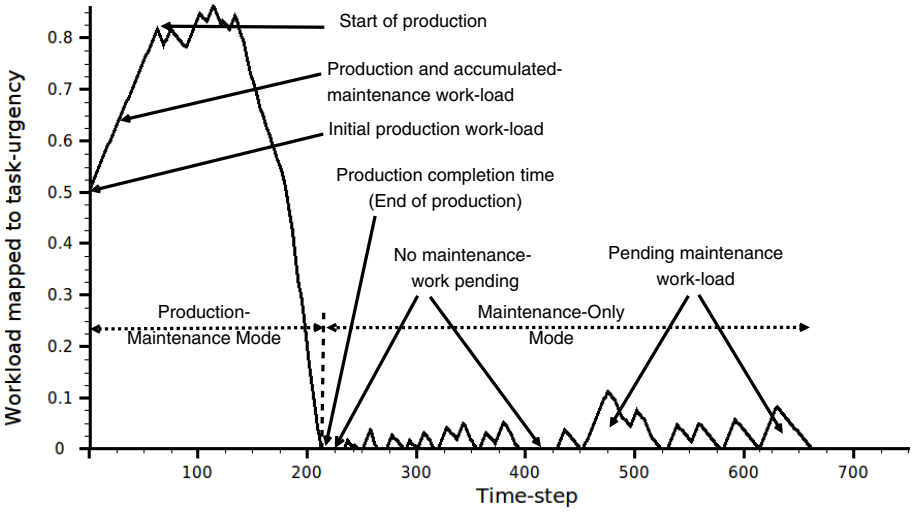


Fig. 3. Manufacturing shop-floor production and maintenance cycles

number of material parts that can be loaded into a machine j in the beginning of its production process and in each time-step, ω_j units of material parts can be processed ($\omega_j \ll W_j$). So let Ω_j^p be the initial production workload of j which is simply: W_j/ω_j unit. We assume that all machines are identical. In each time step, each machine always requires a minimum threshold number of robots, called hereafter as *minimum robots per machine (μ)*, to meet its constant maintenance work-load, Ω_j^m unit. However, if μ or more robots are present in a machine for production purpose, we assume that, no extra robot is required to do its maintenance work separately. These robots, along with their production jobs, can do necessary maintenance works concurrently. For the sake of simplicity, in this paper we consider $\mu = 1$.

Now let us fit the above production and maintenance work-loads and task performance of robots into a unit task-urgency scale. Let us divide our manufacturing operation into two subsequent stages: 1) *production and maintenance mode (PMM)*, and 2) *maintenance only mode (MOM)*. Initially a machine starts working in PMM and does production and maintenance works concurrently. When there is no production work left, it then enters into MOM. Fig. 3 illustrates this for a single machine. Under both modes, let α_j be the amount of workload occurs in a unit time-step if no robot serves a task and it corresponds to a fixed task-urgency $\Delta\phi_{INC}$. On the other hand, let us assume that in each time-step, a robot, i, can decrease a constant workload β_i by doing some maintenance work along with doing any available production work. This corresponds to a negative task urgency: $-\Delta\phi_{DEC}$. So, at the beginning of production process, task-urgency, occurred in a machine due to its production work-loads, can be encoded by Eq. 3.

$$\Phi_{j,INIT}^{PMM} = \Omega_j^p \times \Delta\phi_{INC} + \phi_j^{m0} \tag{3}$$

Here ϕ_j^{m0} represents the task-urgency due to any initial maintenance work-load of j. Now if no robot attends to serve a machine, at each time-step a constant maintenance workload of α_j^m will be added to j and that will increase its task-urgency by $\Delta\phi_{INC}$. So, if k time steps passes without any production work being done, task urgency at k^{th} time-step will follow Eq. 4.

$$\Phi_{j,k}^{PMM} = \Phi_{j,INIT}^{PMM} + k \times \Delta\phi_{INC} \tag{4}$$

However, if a robot attends to a machine and does some production works from it, there would be no extra maintenance work as we assumed that $\mu = 1$. Rather, the task-urgency on this machine will decrease by $\Delta\phi_{DEC}$ amount. If ν_k robots work on a machine simultaneously at time-step k, this decrease will be: $\nu_k \times \Delta\phi_{DEC}$. So in such cases, task-urgency in $(k+1)^{th}$ time-step can be represented by:

$$\Phi_{j,k+1}^{PMM} = \Phi_{j,k}^{PMM} - \nu_k \times \Delta\phi_{DEC} \tag{5}$$

At a particular machine j, once $\Phi_{j,k}^{PMM}$ reaches to zero, we can say that there is no more production work left and this time-step k can give us the *production completion time* of j, T_j^{PMM}. Average production time-steps of a shop-floor with M machines can be calculated by the following simple equation.

$$T_{avg}^{PMM} = \frac{1}{M} \sum_{j=1}^{M} T_j^{PMM} \tag{6}$$

T_{avg}^{PMM} can be compared with the minimum number of time-steps necessary to finish production works, T_{min}^{PMM}. This can only happen in an ideal case where all robots work for production without any random walking or failure. We can get T_{min}^{PMM} from the total amount of work load and maximum possible inputs from all robots. If there are M machines and N robots, each machine has Φ_{INIT}^{PMM} task-urgency, and at each time-step, robots can decrease N $\times$ $\Delta\phi_{DEC}$ task-urgencies, then the theoretical T_{min}^{PMM} can be found from the following Eq. 7.

$$T_{min}^{PMM} = \frac{M \times \Phi_{INIT}^{PMM}}{N \times \Delta\phi_{DEC}} \tag{7}$$

$$\zeta_{avg}^{PMM} = \frac{T_{avg}^{PMM} - T_{min}^{PMM}}{T_{min}^{PMM}} \tag{8}$$

Thus we can define ζ_{avg}^{PMM}, *average production completion delay* (APCD) by following Eq. 8: When a machine enters into MOM, only μ robots are required to do its maintenance works in each time step. So, in such cases, if no robot serves a machine, the growth of task-urgency will follow Eq. 4. However, if ν_k robots are serving this machine at a particular time-step k^{th} , task-urgency at $(k+1)^{th}$ time-step can be represented by:

$$\Phi_{j,k+1}^{MOM} = \Phi_{j,k}^{MOM} - (\nu_k - \mu) \times \Delta\phi_{DEC} \tag{9}$$

By considering $\mu = 1$ Eq. 9 will reduces to Eq. 5. Here, $\Phi_{j,k+1}^{MOM}$ will correspond to the *pending maintenance work-load (PMW)* of a particular machine at a given

time. This happens due to the random task switching of robots with a no-task option (random-walking). Interestingly, PMW will indicate the robustness of this system since higher PMW value will indicate the delay in attending maintenance works by robots. We can find the average PMW (APMW) per machine per time-step, χ_j^{MOM} (Eq. 10) and average PMW for all machines per time-step, χ_{avg}^{MOM} (Eq. 11).

$$\chi_j^{MOM} = \frac{1}{K}\sum_{k=1}^{K} \Phi_{j,k}^{MOM} \qquad (10) \qquad\qquad \chi_{avg}^{MOM} = \frac{1}{M}\sum_{j=1}^{M} \chi_j^{MOM} \qquad (11)$$

3 Implementation

3.1 Design of Our Communication System

As shown in Fig. 2, in this model there exists a centralized *TaskServer* that is responsible for disseminating task information to robots. The contents of task information can be physical locations of tasks, their urgencies and so on. TaskServer delivers this information by emitting *TaskInfo* signals periodically. For example, in a wireless network it can be a message broadcast. Task-Server has another interface for catching feedback signals from robots. The *RobotStatus* signal can be used to inform TaskServer about a robot's current task id, its device status and so on. TaskServer uses this information to update relevant part of task information such as, task-urgency. This up-to-date information is sent in next TaskInfo signal.

3.2 Our Current Implementation

The major components of our implementation are a multi-robot tracking system, robot controller clients and a centralized task-server. In order to track all robots

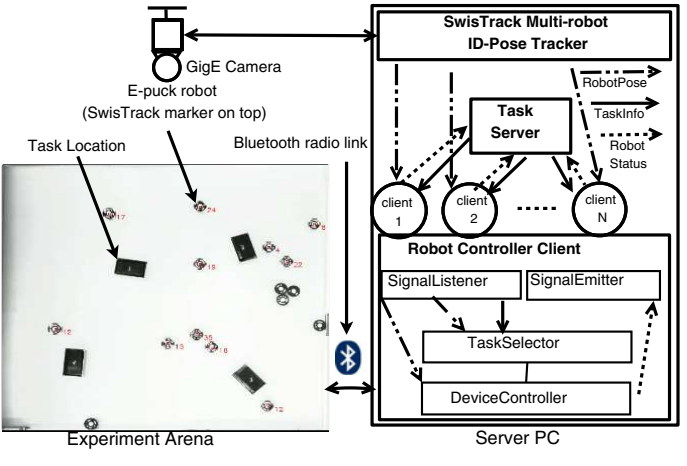


Fig. 4. Hardware and software setup

in real-time, we have used SwisTrack [7], a state of the art open-source, multi-agent tracking system, with a 16-megapixel overhead GigE camera. This set-up gives us the position, heading and id of each of the robots by processing the image frames at about 1 FPS. The interaction of the hardware and software of our system is illustrated in Fig. 4. For inter-process communication (IPC), we have used D-Bus technology[1]. We have developed an IPC component for SwisTrack that can broadcast id and pose of all robots in real-time over our server's D-Bus interface.

Apart from SwisTrack, we have implemented two major software modules: *TaskServer* and *Robot Controller Client (RCC)*. They are developed in Python with its state of the art *Multiprocessing*[2] module. This python module simplifies our need to manage data sharing and synchronization among different sub-processes. As shown in Fig. 4, RCC consists of four sub-processes. *SignalListener* and *SignalEmitter*, interface with SwisTrack D-Bus Server and TaskServer respectively. *TaskSelector* implements AFM algorithms for task selection . *DeviceController* moves a robot to a target task. Bluetooth radio link is used as a communication medium between a RCC and a corresponding e-puck robot.

4 Experiment Design

In this section, we have described the design of parameters and observables of our experiments within the context of our manufacturing shop-floor scenario. These experiments are designed to validate AFM by testing the presence of division of labour, e.g. task specialization, dynamic task-switching or plasticity.

4.1 Parameters

Table 1 lists a set of essential parameters of our experiments. The initial values of task urgencies correspond to 100 units of production work-load without any maintenance work-load as outlined in Eq. 3. For task-urgency values, we choose a limit of 0 and 1, where 0 means no urgency and 1 means maximum urgency. Same applies to sensitisation as well, where 0 means no sensitisation and 1 means maximum sensitisation. The following relationships are maintained for selecting task-urgency and sensitization parameters.

$$\Delta\phi_{INC} = \frac{\Delta\phi_{DEC} \times N}{2 \times M} \tag{12}$$

$$\Delta k_{DEC} = \frac{\Delta k_{INC}}{M - 1} \tag{13}$$

Eq. 12 establishes the fact that task urgency will increase at a higher rate than that of its decrease. As we do not like to keep a task left unattended for a long time we choose a higher rate of increase of task urgency. This difference is set on the basis of our assumption that at least half of the expected number of robots

[1] http://dbus.freedesktop.org/doc/dbus-specification.html
[2] http://docs.python.org/library/multiprocessing.html

Table 1. Experimental parameters

Parameter	Value
Initial production work-load/machine (Ω_j^p)	100 unit
Task urgency increase rate ($\Delta\phi_{INC}$)	0.005
Task urgency decrease rate ($\Delta\phi_{DEC}$)	0.0025
Initial sensitization (K_{INIT})	0.1
Sensitization increase rate (Δk_{INC})	0.03
Sensitization decrease rate (Δk_{DEC})	0.01

(ratio of number of robots to tasks) would be available to work on a task. So they would produce similar types of increase and decrease behaviours in task urgencies. Eq. 13 suggests that the learning will happen much faster than the forgetting. The difference in these two rates is based on the fact that faster leaning gives a robot more chances to select a task in next time-step and thus it becomes more specialized on it. Task-Server updates task-info messages in the interval of 5s and robots stick on to a particular task for a maximum of 10s.

4.2 Observables

We have defined a set of observables to evaluate our implementation. The first two observables, the changes in task-urgencies and the changes in active worker ratios, can give us an overall view of plasticity of division labour. Our third observable is to find the changes in robot task specialization which is also an important measure of division of labour. Our last measurement is the communication load which is specific to this particular implementation and corresponds to the continuous flow of information. Within the context of our manufacturing shop-floor scenario, we measure the average production completion delay (APCD) and average pending maintenance work (APMW) as the metrics of manufacturing shop-floor performance.

5 Results and Discussions

Our AFM validation experiments were conducted with 16 robots, 4 tasks in an arena of 4 m^2 for about 40 minutes and averaged them over five iterations. Fig. 5 shows the dynamic changes in task urgencies in one iteration. In order to describe our system's dynamic behaviour holistically, we analyse the changes in task urgencies over time. Let $\phi_{j,q}$ be the urgency of a task j at q^{th} time-step and $\phi_{j,q+1}$ be the task urgency of $(q+1)^{th}$ time-step. We can calculate the sum of changes in urgencies of all tasks at $(q+1)^{th}$ time-step:

$$\Delta\Phi_{j,q+1} = \sum_{j=1}^{M} (\phi_{j,q+1} - \phi_{j,q}) \tag{14}$$

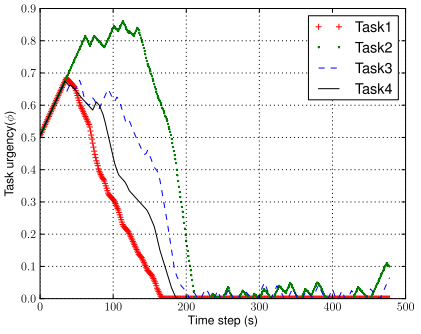


Fig. 5. Dynamic task-urgency changes

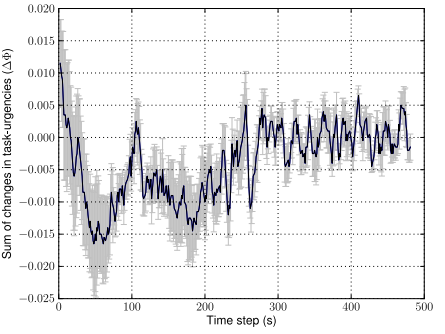


Fig. 6. Shop-floor workload history

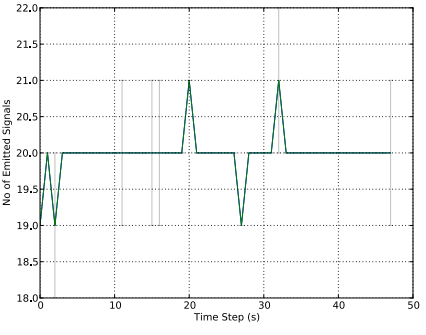


Fig. 7. Task server's task-info broadcasts

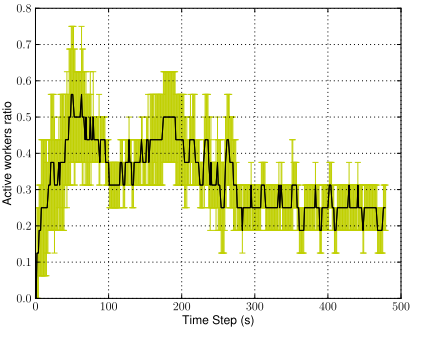


Fig. 8. Self-organized task-allocation

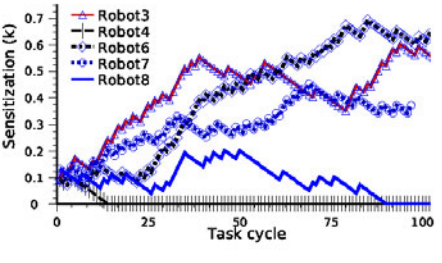


Fig. 9. Task specialization on Task3

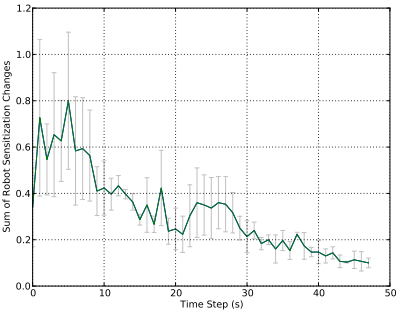


Fig. 10. Changes in sensitizations

From Fig. 6 we can see that initially the sum of changes of task urgencies are towards negative direction. This implies that tasks are being served by a high number of robots. Fig. 8 shows that in production stage, when work-load is high, many robots are active in tasks and this ratio varies according to task urgency changes. Fig. 9 gives us the task specialization of five robots on Task3 in a particular run of our experiment. This shows us how our robots can specialize (learn) and de-specialize (forget) tasks over time. The de-specialization of tasks is calculated similar to Eq. 14. We have calculated the absolute sum of changes in sensitizations of all robots by the following equation.

$$\Delta K_{j,q+1} = \sum_{j=1}^{M} |(k_{j,q+1} - k_{j,q})| \tag{15}$$

This values of ΔK are plotted in Fig. 10. It shows that the overall rate of learning decreases and forgetting increases over time. It is a consequence of the gradually increased task specialization of robots and reduced task-urgencies over time. Fig. 7 presents the frequency of signalling task information by TaskServer. Since the duration of each time step is 50s long and TaskServer emits signal in every 2.5s, there is an average of 20 signals in each time-step.

Within our manufacturing shop-floor scenario, we have got average production completion time 165 time-steps (825s) where sample size is (5 x 4) = 20 tasks, SD = 72 time-steps (360s). According to Eq. 7, our theoretical minimum production completion time is 50 time-steps (250s) assuming the non-stop task performance of all 16 robots with an initial task urgency of 0.5 for all 4 tasks and task urgency decrease rate $\Delta\Phi_{DEC} = 0.0025$ per robot per time-step. Hence, Eq. 8 gives us APCD, $\zeta = 2.3$ which means that our system has taken 2.3 times more time (575s) than the estimated minimum time. Besides, from the average 315 time-steps (1575s) maintenance activity of our robots per experiment run, we have got APMW, $\chi = 0.012756$ which corresponds to the pending work of 3 time-steps (15s) with sample-size = 20 tasks, SD = 13 time-steps (65s), where $\Delta\Phi_{INC} = 0.005$ per task per time-step. This tells us the robust task performance of our robots which can return to an abandoned task within a minute or so.

6 Conclusion and Future Work

In this paper we have validated an inter-disciplinary generic model of self-regulated division of labour or multi-robot task allocation by incorporating it in our multi-robot system under a manufacturing shop-floor scenario. A centralized communication system has been instantiated to realize this model. We have evaluated various aspects of this model, such as the ability to meet dynamic task demands, individual robot's task specializations, system-wide communication loads and flexibility in concurrent task completions. A set of metrics has been proposed to observe the division of labour in this system. From our experimental results, we have found that our attractive filed model can meet the requirements of dynamic division of labour by the virtue of its self-regulatory

behaviours. In the future, we plan to extend this work using various local peer-to-peer communication models in a multi-robot system having about 40 e-puck robots.

Acknowledgements. This research has been funded by the Engineering and Physical Sciences Research Council (EPSRC), UK, grant reference EP/E061915/1.

References

1. Arcaute, E., Christensen, K., Sendova-Franks, A., Dahl, T., Espinosa, A., Jensen, H.J.: Division of labour in ant colonies in terms of attractive fields. In: Ecol. Complex (2008)
2. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: from natural to artificial systems. Oxford University Press, Oxford (1999)
3. Dias, M.B., Zlot, R.M., Kalra, N., Stentz, A.: Market-based multirobot coordination: A survey and analysis. Proceedings of the IEEE 94, 1257–1270 (2006)
4. Gerkey, B.P., Mataric, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. The International Journal of Robotics Research 23, 939 (2004)
5. Lerman, K., Jones, C., Galstyan, A., Mataric, M.J.: Analysis of dynamic task allocation in multi-robot systems. The International Journal of Robotics Research 25, 225 (2006)
6. Liu, W., Winfield, A.F.T., Sa, J., Chen, J., Dou, L.: Towards energy optimization: Emergent task allocation in a swarm of foraging robots. Adaptive Behavior 15(3), 289–305 (2007)
7. Lochmatter, T., Roduit, P., Cianci, C., Correll, N., Jacot, J., Martinoli, A.: Swistrack-a flexible open source tracking software for multi-agent systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2008, pp. 4004–4010 (2008)
8. Parker, L.E.: Distributed intelligence: Overview of the field and its application in multi-robot systems. Journal of Physical Agents, special issue on multi-robot systems 2(2), 5–14 (2008)
9. Shen, W., Norrie, D.H., Barthes, J.P.: Multi-agent systems for concurrent intelligent design and manufacturing. Taylor & Francis, London (2001)

A Thermodynamic Approach to the Analysis of Multi-robot Cooperative Localization under Independent Errors

Yotam Elor and Alfred M. Bruckstein

Faculty of Computer Science and the Goldstein UAV and Satellite Center, Israel
{yotame,freddy}@cs.technion.ac.il

Abstract. We propose a new approach to the simultaneous cooperative localization of a group of robots capable of sensing their own motion on the plane and the relative position of nearby robots. In the last decade, the use of distributed optimal Kalman filters (KF) to solve this problem have been studied extensively. In this paper, we propose to use a sub-optimal Kalman filter (denoted by EA). EA requires significantly less computation and communication resources then KF. Furthermore, in some cases, EA provides better localization.

In this paper EA is analyzed in a soft "thermodynamic" fashion i.e. relaxing assumptions are used during the analysis. The goal is not to derive hard lower or upper bounds but rather to characterize the robots expected behavior. In particular, to predict the expected localization error. The predictions were validated using simulations. We believe that this kind of analysis can be beneficial in many other cases.

1 Introduction

Localization is the task of estimating the robot location. It has been identified as one of the key problems in robotics. The localization problem can be roughly divided into two variants: In the first variant the robots can estimate their location by sensing their surroundings and comparing it with knowledge they have regarding the environment (e.g. a map). While in the second variant, the robots have no such capabilities. Instead, every robot knows its initial location and updates its location estimation based on odometry readings.

There is a vast body of literature discussing the first variant where the main challenge is to incorporate the large quantity of data gathered by the robot (or robots) into a consisted world view. The means to gather data are the *exteroceptive* sensors which surveys the world and the *proprioceptive* sensors (odometry) which monitor the robot itself e.g. GPS, compass, wheel encoders, etc. This paper's focus is on the second variant, the reader interested in the more general scenario is referred to the book by Borenstein *et al.* [1] and the survey of Thrun [12].

In the second variant of the localization problem it is assumed that, initially, every robot knows its location and uses odometry in order to track

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 36–47, 2010.

it (dead-reckoning). However, due to noisy sensor readings, in time, the estimation diverges from the robot real location. When a group of robots perform localization, the localization error can be reduced by sharing information between them. In order to do so, several *exteroceptive* capabilities are needed i.e. every robot is required to be able to sense the relative location of nearby robots and to communicate with them.

We denote the cooperative localization algorithm proposed in this is paper by "Error Averaging" (EA). In EA, every robot moves in the area while maintaining an estimate of its location using its odometry. Whenever two robots are within sensing and communication range, they average their location estimations. In case the two robots' localization errors are uncorrelated, such an averaging will result in reducing the error of both robots. In this work, a large group of identical robots performing EA is considered. It is natural to apply thermodynamic approach when considering such a large and homogeneous group. This paper's goal is to characterize the behavior of the robots. In particular, we are interested in predicting the expected localization error. The goal is not to derive hard lower or upper bounds but rather to apply relaxing assumptions during the analysis in order to predict the expected behavior.

A very simple "independent error" model (IEM) is considered. In IEM, the odometry errors are independent of the state of the robot. The localization error is accumulated as a two-dimension Gaussian. Furthermore, the errors added at different times are statistically independent. This model is relevant, for example, when considering a small flying platform in an indoor environment (no wind). In this case, the odometry errors are due to small air currents which are independent of the flying platform actions.

Due to space limitations, some details are ommited from this paper, they can be found in our readily available TR [2].

2 Model

The notations of [11] are adopted when possible. A group of M identical independent robots is considered. The robots move in a flat environment of size A. Every robot can communicate with its neighbors up to a constant limited distance V. The robots have sensors which enable them to detect nearby robots and sense their relative location (up to distance V). In order to keep the model simple, it is assumed that these sensors are error free i.e. the robots are able to sense the relative location of each other accurately. Extending our formulation to include noisy *exteroceptive* sensors is straightforward.

Two robots "meet" when the distance between them is at most V i.e. they can sense each other and communicate. Upon meeting, the robots average their location estimations hence reducing the localization error. So the frequency of meetings strongly effect the localization quality. Roughly speaking, the higher the frequency of meetings the lower the localization error. The frequency of meetings is determined by the movement pattern of the robots which in turn is application dependent. In this work it is assumed that the robots perform

the following random walk: All robots travel at constant speed and the heading of every robot is generally fixed. However, upon hitting an obstacle, the robot randomize a new heading.

Discrete time is considered i.e. $t = 0, 1, 2....$ There are M robots modeled as points on the plane. The location of robot r_i in respect to a fixed reference frame is denoted by the vector $X_i(t) = [x_i(t), y_i(t), \phi_i(t)]^T$ where $x_i(t)$, $y_i(t)$ are the robot's coordinates and $\phi_i(t)$ is the robot's heading. Let v_0 be the (constant) robot speed and $\omega_i(t)$ its angular velocity at time t. The robot coordinates are updated in the normal way i.e.,

$$X_i(t+1) = X_i(t) + \begin{bmatrix} \cos(\phi_i(t)) & 0 \\ \sin(\phi_i(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_0 \\ \omega_i(t) \end{bmatrix} \tag{1}$$

The location estimation of robot r_i at time t is denoted by $\hat{X}_i(t) = \left[\hat{x}_i(t), \hat{y}_i(t), \hat{\phi}_i(t)\right]^T$. Initially $\hat{X}(0) = X(0)$. The estimation error is given by the vector $\tilde{X} = \hat{X} - X = \left[\tilde{x}_i(t), \tilde{y}_i(t), \tilde{\phi}_i(t)\right]^T$.

$z \sim N\left(0, \sigma^2\right)$ implies that z is a random variable distributed normally in one-dimension with zero mean and variance of σ^2. For practical reasons we are interested in the expected error i.e. the expected distance between the robot location to where the robot thinks it is. The expected error for robot r_i is the expected value of $e_i = \sqrt{\tilde{x}_i^2(t) + \tilde{y}_i^2(t)}$ and is denoted by $E[e_i]$. In case $\tilde{x}_i \sim N\left(0, \sigma^2\right)$ and $\tilde{y}_i \sim N\left(0, \sigma^2\right)$, $E[e_i] = \sqrt{\frac{\pi}{2}\sigma^2}$.

In IEM, the localization errors added at each time step are independent of the robot state. It is assumed that the errors are distributed normally i.e.,

$$\hat{X}_i(t+1) = \hat{X}_i(t) + \begin{bmatrix} \cos(\phi_i(t)) & 0 \\ \sin(\phi_i(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_0 \\ \omega_i(t) \end{bmatrix} + \begin{bmatrix} N_{i,x}(t) \\ N_{i,y}(t) \\ 0 \end{bmatrix} \tag{2}$$

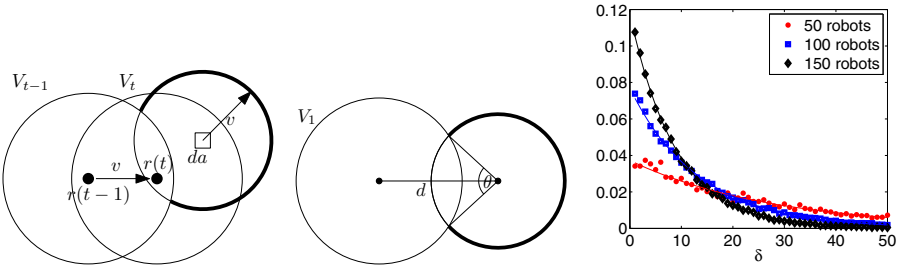
Where $N_{i,x}(t) \sim N\left(0, \sigma_0^2\right)$ $\left(N_{i,y}(t) \sim N\left(0, \sigma_0^2\right)\right)$ is the noise added to $\hat{x}_i$ $(\hat{y}_i)$ at time t and σ_0 is a constant. The errors are independent i.e. for any $i \neq j$ or $a \neq b$ (where $a, b \in \{x, y\}$) or $t_1 \neq t_2$: $N_{i,a}(t_1)$ and $N_{j,b}(t_2)$ are statistically independent. Equation 2 can be written in the following manner

$$\hat{X}_i(t) = X_i(t) + \sum_{t'=0}^{t-1} \left[N_{i,x}(t')\ N_{i,y}(t')\ 0\right]^T \tag{3}$$

Hence, $\tilde{X}_i \sim \left[N\left(0, t \cdot \sigma_0^2\right)\ N\left(0, t \cdot \sigma_0^2\right)\ 0\right]^T$. So when no localization correction mechanisms are applied, the variance of the localization error grows linearly in time.

3 Average Free Path

In this section we will investigate the "average free path" i.e. the average distance traveled by a robot between meetings. Fix a robot r. Let p_r be the probability of r



(a) (b) (c) The histogram of δ

Fig. 1. Deriving and measuring p_r

meeting any other robot at time t. Denote by $r(t)$ the position of robot r at time t. Let V_{t-1} be the circle of radius V around $r(t-1)$ and V_t - the circle around $r(t)$, see Figure 1a. So robot r meets robot r' at time t if and only if $r'(t) \in V_t$ and $r'(t-1) \notin V_{t-1}$. Note that in case $r'(t) \in V_t$ and $r'(t-1) \in V_{t-1}$ the distance between r and r' at time t is less than V. However, in this case, further exchange of information between the robots will not improve their localization (as shown in Section 4). So it is not considered as a meeting.

To derive the probability that there is a robot r' such as $r'(t) \in V_t$ and $r'(t-1) \notin V_{t-1}$, let

$$p_{r'} = \int_{V_t} Pr\left[r'(t) \in da\right] \cdot Pr\left[r'(t-1) \notin V_{t-1} \,|\, r'(t) \in da\right] da \tag{4}$$

Assuming the robots are distributed uniformly over the area, $Pr\left[r'(t) \in da\right] = da/A$. In case $r'(t) \in da$, it is assumed that $r'(t-1)$ is distributed uniformly over the circle of radius v_0 (the step length) with a center at $r'(t)$. So $Pr[r'(t-1) \notin V_{t-1} \,|\, r'(t) \in da]$ equals the part of this circle which is not inside V_0 (the bold arc in Figure 1a). Let $\Delta x = x_{r'}(t) - x_r(t)$, $\Delta y = y_{r'}(t) - y_r(t)$. We face the problem of calculating the intersection of two circles at distance $d = \sqrt{(\Delta x + v_0)^2 + \Delta y^2}$, see Figure 1b. θ is given by:

$$\theta = \begin{cases} 2\text{asin}\left(\frac{\sqrt{4d^2V^2 - \left(d^2 - v_0^2 + V^2\right)^2}}{2dv_0}\right) & \text{if } r'(t) \notin V_{t-1} \\ 2\pi - 2\text{asin}\left(\frac{\sqrt{4d^2V^2 - \left(d^2 - v_0^2 + V^2\right)^2}}{2dv_0}\right) & \text{if } r'(t) \in V_{t-1} \end{cases}.$$

So $Pr\left[r'(t-1) \notin V_{t-1} \,|\, r'(t) \in da\right] = \frac{2\pi - \theta}{2\pi}$ and finally $p_{r'} = \int_{V_t} \frac{2\pi - \theta}{2\pi A} da$. $p_{r'}$ can be calculated numerically. The probability that robot r meets any other robot at time t is given by $p_r = 1 - (1 - p_{r'})^{n-1}$.

Let δ be the time elapsed between two successive meetings of a specific robot with any other robot i.e. δ is the "free time" between meetings. δ is distributed

geometrically with a mean of $1/p_r$, i.e. $Pr[\delta = k] = p_r(1-p_r)^{k-1}$. p_r can not be measured experimentally. Instead, the distribution of δ was measured. The histogram of δ for three group sizes is presented in Figure 1c, where the solid lines are the theoretical predictions and the dots are the simulation results. In those three runs the environment was a torus of size 100×100, $V = 3$ and $v_0 = 1$. Every line in the figure is a result of a single run of 5000 time steps. The experiments show that the estimation of δ (hence p_r) is very accurate.

4 The Covariance Matrix

In IEM, the components of X_i are independent so they can be analyzed separately. Hence only x component will be analyzed. The results apply to y as well. Let $P_x(t)$ be the covariance matrix of the localization errors of x at time t. Where the components of $P_x(t)$ are denoted by $\sigma_{i,j}(t)$ and given by $\sigma_{ij}(t) = Cov[\tilde{x}_i(t), \tilde{x}_j(t)]$.

We would like to examine the evolution of P_x in time. Considering a group of robots performing EA, $P_x(t)$ can be derived from $P_x(t-1)$ in two stages. In the first - an error is added and in the second - meetings between robots are accounted for. The error addition stage is given by $P_x(t^-) = P_x(t-1) + I_M \cdot \sigma_0^2$ where I_M is the unit matrix of size M. To account for meetings, consider all pairs of robots which met at time t. Every meeting is considered separately[1]. Let r_i, r_j be two robots who have met at time t. The meeting process is described for robot r_i; r_j follows the same procedure in parallel. Upon meeting, r_i asks r_j "what is your estimation of my location?". r_j replies with $\hat{x}_j(t^-) + (x_i(t) - x_j(t))$. Then, r_i sets its location estimation to the average of its previous estimation and the coordinates received from r_j, i.e.

$$\hat{x}_i(t) = \frac{\hat{x}_i(t^-) + (\hat{x}_j(t^-) + x_i(t) - x_j(t))}{2} \tag{5}$$

$$\tilde{x}_i(t) = \frac{\tilde{x}_i(t^-) + \tilde{x}_j(t^-)}{2} \sim N\left(0, \frac{\sigma_i^2(t^-) + \sigma_j^2(t^-) + 2\sigma_{ij}(t^-)}{4}\right) \tag{6}$$

To gain insight of Equation 6, observe two limit cases. In case $\sigma_i^2(t^-) = \sigma_j^2(t^-)$ and $\tilde{x}_i(t^-)$, $\tilde{x}_j(t^-)$ are independent (i.e. $\sigma_{ij}(t^-) = 0$), $\sigma_i^2(t) = \sigma_i^2(t^-)/2$ i.e. the variance was halved. In case $\sigma_i^2(t^-) = \sigma_j^2(t^-)$ and $\tilde{x}_i(t^-)$, $\tilde{x}_j(t^-)$ are fully correlated (i.e. $\sigma_{ij}(t^-) = \sigma_i^2(t^-)$), $\sigma_i^2(t) = \sigma_i^2(t^-)$ i.e. the localization was not improved. Note that after the update $\tilde{x}_i(t) = \tilde{x}_j(t)$ so $\tilde{x}_1(t)$ and $\tilde{x}_2(t)$ are fully correlated and another averaging of their location estimations will not reduce the error. To conclude, after a meeting between r_i and r_j

$$\sigma_i^2(t) = \sigma_j^2(t) = \sigma_{ij}(t) = \frac{\sigma_i^2(t^-) + \sigma_j^2(t^-) + 2\sigma_{ij}(t^-)}{4} \tag{7}$$

[1] In case robot r have met two robots in the same time cycle. It is assumed that r first average its localization with one of them and afterward - with the other.

Note that other elements of P_x must be updated as well. Consider any robot r_k $(k \neq i, j)$,

$$\sigma_{ik}(t) = E\left[\tilde{x}_i(t)\,\tilde{x}_k(t)\right] = E\left[\frac{\tilde{x}_i(t) + \tilde{x}_j(t)}{2}\tilde{x}_k(t)\right] = \frac{\sigma_{ik}(t^-) + \sigma_{jk}(t^-)}{2} \quad (8)$$

So $\sigma_{ik}(t)$ equals the average of $\sigma_{ik}(t^-)$ and $\sigma_{jk}(t^-)$. Note that the total sum of P_x does not change as a result of meetings so

$$E\left[P_x(t)\right] = M \cdot \sigma_0^2 \cdot t/M^2 = \frac{\sigma_0^2}{M} \cdot t \quad (9)$$

So EA does not reduce the total amount of noise in P_x. Nevertheless, EA spreads the error from the main diagonal of P_x to the rest of the matrix. Since the robots' localization error is proportional only to the values of the main diagonal of P_x, spreading the error is desired.

The evolution of P_x, when the robots follow EA, can be described as follows. Initially: $P_x = 0$. Then, as the robots move, the values on the main diagonal of P_x starts growing. Due to meetings between robots, error from the main diagonal spread to the rest of P_x. Finally, a semi-steady state is achieved in which the rate of removing error from the main diagonal (almost) equals the rate of adding error to it. The analysis purpose is to predict the characteristics of such a semi-steady state. This is achieved by assuming that P_x comprises only two values: $\sigma_{diag}^2(t)$ and $\sigma_{cov}^2(t)$. It is assumed that for any i, $\sigma_i^2(t) = \sigma_{diag}^2(t)$ and for any $i \neq j$, $\sigma_{ij}(t) = \sigma_{cov}^2(t)$. The two latter assumptions are clearly false (P_x comprises many values). However, when considering a large group of homogeneous robots, it is a reasonable approximation to assume that the localization uncertainties related to most robots (the values of the main diagonal of P_x) are about the same. In the same spirit, it is reasonable to assume that the covariances between most robots are about the same.

By neglecting the main diagonal contribution to $E\left[P_x\right]$, $\sigma_{cov}^2(t)$ can be approximated by

$$\sigma_{cov}^2(t) \simeq E\left[P_x(t)\right] = \frac{\sigma_0^2}{M} \cdot t \quad (10)$$

Consider any robot r_i. Let p_r be the probability that r_i meets another robot at time t. In case r_i does not meet another robot at time t,

$$\sigma_i^2(t+1) = \sigma_i^2(t) + \sigma_0^2 \quad (11)$$

In case r_i meets r_j at time t,

$$\sigma_i^2(t+1) = \frac{\sigma_i^2(t) + \sigma_j^2(t) + 2\sigma_0^2 + 2\sigma_{ij}(t)}{4} \quad (12)$$

Using $\sigma_i^2 \simeq \sigma_j^2 \simeq \sigma_{diag}^2$, $\sigma_{ij} \simeq \sigma_{cov}$ and Equations 11 and 12 we get

$$\sigma_{diag}^2(t+1) = p_r \frac{\sigma_{diag}^2(t) + \sigma_0^2 + \sigma_{cov}^2(t)}{2} + (1 - p_r)\left(\sigma_{diag}^2(t) + \sigma_0^2\right) \quad (13)$$

Equations 10 and 13 form a set of two difference equations. The fixed point solution for this set is given by

$$\sigma^2_{cov}(t) = \frac{\sigma_0^2}{M} \cdot t \qquad \sigma^2_{diag}(t) \simeq \frac{2\sigma_0^2}{p_r} + \frac{\sigma_0^2}{M} \cdot t \tag{14}$$

So the components of P_x not on the main diagonal grows linearly in time at a rate of σ_0^2/M. The values on the main diagonal grow with the same rate. However, there is a constant gap of about $2\sigma_0^2/p_r$ between the values on main diagonal and the rest of the covariance matrix. This constant component is a result of the time the odometry errors require to average over the robots. Note that the error remains unbounded.

4.1 Using a Landmark

Consider a landmark placed in a fixed point in the environment. The robots know the exact coordinates of the landmark and every robot that is within the landmark sensing range updates its localization accordingly. As a consequence, every time a robot sense the landmark, it's localization error is zeroed. Furthermore, since its new localization is uncorrelated with the other robots, the covariance of the robot with all other robots is also reduced to zero. Note that any robot can be used as a landmark. It just needs to stay put and report its coordinates to every robot in its communication range. By applying a formalism similar to the one used in the case without a beacon we get,

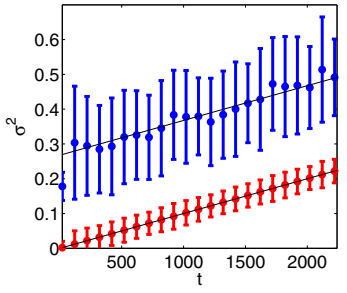
$$\sigma^2_{cov} \simeq \frac{\sigma_0^2}{2Mp_l} \qquad \sigma^2_{diag} \simeq \frac{2\sigma_0^2}{p_r} + \sigma^2_{cov} \tag{15}$$

where p_l is the probability of sensing the landmark and is derived similarly to p_r. The full process of deriving the equations above is given in our TR [2]. It is important to note that a single landmark is sufficient to make the localization error bounded.

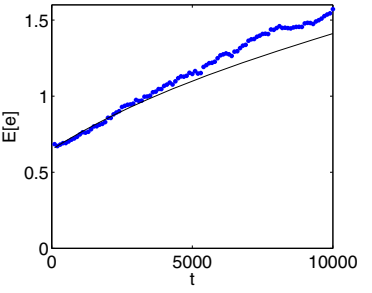
5 Discussion and Simulations

Simulations were used in order to validate the analytical results. In all figures, the solid lines are the theoretical estimations and the error bars (or markers) represent the simulation results. When presenting σ^2_{diag}, the average of the main diagonal of P_x is presented with its standard deviation. When σ^2_{cov} is presented, the average of P_x without the main diagonal is presented.

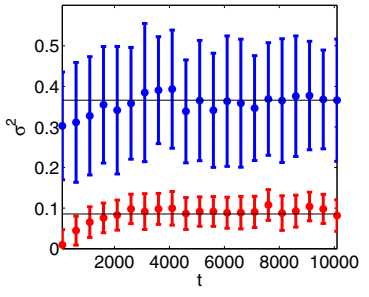
The values of σ^2_{cov}, σ^2_{diag} for a single run on a torus are presented in Figure 2a where $M = 100$, $A = 100^2$, $V = 3$, $v_0 = 1$ and $\sigma_0^2 = 0.01$. The experiments show that the estimations of σ^2_{diag} and σ^2_{cov} are very accurate. The mean error was found to be very noisy for a single run. Hence the average of the mean error over 50 runs is presented in Figure 2b. The average is less noisy and it can be observed that the mean error is predicted well. The values of σ^2_{cov}, σ^2_{diag} for a single run on a torus with a landmark are presented in Figure 2c. The average



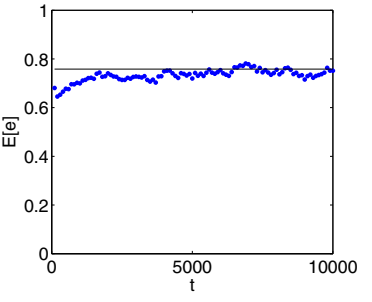
(a) The lower line is σ^2_{cov} and the upper is σ^2_{diag}



(b) Average of the mean error over 50 runs



(c) The lower line is σ^2_{cov} and the upper is σ^2_{diag}



(d) Average of the mean error over 50 runs

Fig. 2. (2a) σ^2_{cov}, σ^2_{diag} for a single run on a torus and (2b) $E[e]$ averaged over 50 runs. (2c) σ^2_{cov}, σ^2_{diag} for a single run on a torus with a landmark and (2d) $E[e]$ averaged over 50 runs.

of the mean error over 50 runs is presented in Figure 2d. The simulation results agree with the analysis.

The simulations have shown that the approximations are accurate when the environment is a torus. Simulation results in environments that include obstacles can be found in our TR [2]. For environments with obstacles, σ^2_{cov} and the growth rate of σ^2_{diag} are predicted well but the constant gap between σ^2_{cov} and σ^2_{diag} is higher then expected. Recall that this gap is the result of the time required to average the error over the robots and is given by $2\sigma_0^2/p_r$. On the torus, the robots travel freely hence at every time step there is a probability of p_r to meet a "fresh" robot i.e. a robot such as the localization covariance with it is σ^2_{cov}. On the contrary, in environment with obstacles (e.g. the environement is divided into "rooms"), there is a high probability to meet a "dirty" robot i.e. a robot with high shared covariance due to a recent meeting. Meeting a "fresh" robot reduces the localization error much more efficiently that meeting a "dirty" one. Putting it another way, for environments with obstacles, p_r is lower than given in Section 3 hence the gap is larger.

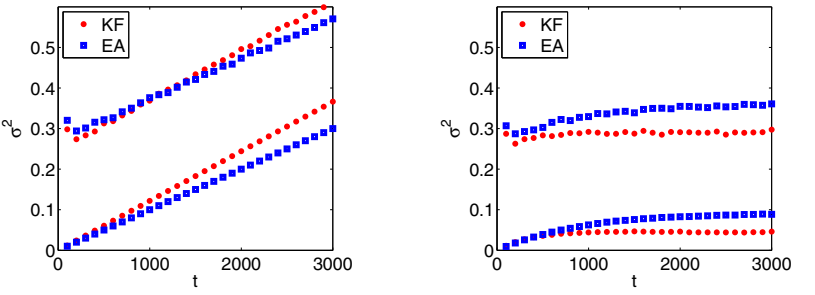
6 Previous Work

About ten years ago, Sanderson [11] have proposed a cooperative localization mechanism almost similar to EA. In his work, an optimal Kalman filter (KF) is used in the localization process i.e. when two robots exchange location information, they perform a weighted average based on their localization quality. The robot with the more precise localization (lower σ value) is given a higher weight. In order to calculate the weights, the meeting robots need to know their covariance. Hence, Sanderson have proposed a central (non-distributed) correction mechanism. He have also presented a distributed algorithm for the fully symmetric case: homogeneous group and a complete relative position measurement graph (RPMG) i.e. at every time step all robots meet all robots. In EA, a suboptimal Kalman filter is applied i.e. the weights of both robots are equal. Hence the robots are not required to maintain the covariance matrix.

Roumeliotis and colleagues [9] have presented a distributed version of KF in which the computation required to maintain the covariance matrix is distributed between the robots. However, every meeting between two robots implies an update of $2M$ components of the covariance matrix. Furthermore, all robots must be aware of every update of the covariance matrix. In the distributed KF of Roumeliotis *et al.*, every meeting implies a computation complexity of $\Theta\left(M^2\right)$ and communication between all robots, so their algorithm does not scale well. In order solve that problem, Mourikis and Roumeliotis have proposed to reduce the computation and communicating loads by lowering the frequency of relative observations [6]. Martinelli have proposed to use hierarchical structure of Kalman filters [5] i.e. the robots are divided into groups, relative observations and corrections are performed within each group, inter-group corrections are performed only between the group leaders hence reducing the computation and communication complexity. On the contrary, in EA, the computation complexity implied by a meeting is $\Theta\left(1\right)$ and the only communication required is between the two meeting robots.

In order to compare between KF and EA, a centralized version of KF was implemented. In KF, when two robots r_i, r_j meet, r_i updates its localization to be $\hat{x}_i\left(t\right)=\alpha\hat{x}_i\left(t^-\right)+(1-\alpha)\left(\hat{x}_j\left(t^-\right)+x_i\left(t\right)-x_j\left(t\right)\right)$, where $\alpha=\frac{\sigma_j^2\left(t^-\right)-\sigma_{ij}\left(t^-\right)}{\sigma_i^2\left(t^-\right)+\sigma_j^2\left(t^-\right)-2\sigma_{ij}\left(t^-\right)}$ is chosen such as to minimize $\tilde{x}_i\left(t\right)$ (compare with Equation 5 in which $\alpha=\frac{1}{2}$).

Observe Figure 3 for experimental comparison between KF and EA. Result on a torus without a landmark are presented in Figure 3a. At the beginning of the process, until $t=500$, KF performs better then EA due to its faster error dispersion rate. However, KF causes the covariance matrix to include more energy then EA (observe the two bottom lines in the figure). Hence, KF reduce the constant component of the localization error but increase the slope of the time dependent component. So, in time, EA outperforms KF. This can be explained as follows. Consider a meeting between two robots r_i and r_j. Assume that the localization of r_i is better then the localization of r_j i.e. $\sigma_i^2<\sigma_j^2$. So when applying KF, the weight given to r_i is higher. The robots are homogeneous, so $\sigma_i^2<\sigma_j^2$ implies that r_i have met more robots prior to its meeting with r_j.



(a) Torus without a landmark (b) Torus with a landmark

Fig. 3. Comparison between EA (blue squares) and KF (red dots) under IEM. The bottom lines correspond to the mean value of the covariance matrix ($E[P_x]$) and the upper - the mean value of the main diagonal of P_x.

Hence r_i shares more covariance with other robots. Upon meeting, these high covariances will be copied to r_j due to r_i's high weight thus contributing to the overall value of the covariance matrix.

Experimental comparison on a torus with a landmark are presented in Figure 3b. The results were averaged over 50 runs where $M = 100$, $V = 3$, $v_0 = 1$ and $\sigma_0^2 = 0.01$. When the environment include a landmark, KF outperforms EA due to its faster error dispersing rate. KF is able to transfer the quality localization from the landmark to the robots more efficiently. Since meetings with the landmark continuously remove energy from the covariance matrix, the bad side effect of KF is negligible.

Roumeliotis and Rekleitis were the first to analyze the performance of KF [10]. They have considered homogeneous robots with complete RPMG. Later, Mourikis and Roumeliotis have extended the analysis to include heterogeneous groups and general RPMG [7]. Mourikis and Roumeliotis have analyzed KF assuming a fixed RPMG i.e. every robot average its location with a fixed set of other robots[2]. By fixing the RPMG, they have been able to obtain an exact analysis of the localization process. In this work, the RPMG is not fixed hence approximations must be used.

The model used by Mourikis and Roumeliotis is more suitable to ground robots then IEM. In their model, every robot sense its orientation using a compass and updates its localization based on the distance and direction traveled. The localization errors result from the wheel encoders and compass noises. So the localization errors added at each time step are independent but are effected by the robot state (heading, speed). We intend to apply our formalizm on more realistic models like the one considered in [7] in the future. Even though the model of Mourikis and Roumeliotis is more general then IEM, the analysis of both models produce similar results. The main similarities are:

[2] They have also considered changes of the RPMG but their results discuss the system state after stabilization.

- The error comprises a time dependent term and a constant term. The time dependent term is monotonically increasing (in time) and is dependent solely on the number of robots and the quality of the odometry. In particular, it is not dependent on the RPMG. Observe Equation 14. The time dependent part is dependent of σ_0^2 (odometry noise) and M but is independent of p_r (a characteristic of the RPMG).
- When a single robot (or more) have access to absolute position measurement, the error of all robots become bounded. In our work, this happens when a landmark is introduced.

With resemblance to KF, Fox *et al.* have proposed to average the location estimations between robots [3]. In their work, every robot estimate its location using Monte Carlo localization [13] i.e. every robot maintains a cloud of points in space with a probability attached to every point. The robot location estimation is the probability function implied by the cloud. When two robots sense each other, their clouds are averaged.

Kurazume and colleagues [4] have proposed a strategy based on "portable landmarks". In this scheme, every time a robot moves, other robots are holding still while following the robot movement with their sensors. The viewing robots supply the moving one with a localization better then given by its own odometry. Later, several other works were carried out using this scheme, see [8]. Since this strategy is not solely odometry based, it is more resilient to correlation between odometry errors. On the downside, when applying this scheme, the robots' movements are limited. Where in EA, no special movement pattern is required, the robots are free to go wherever the task they perform requires.

7 Conclusion

We have presented the error averaging (EA) localization scheme inspired by the optimal Kalman filter (KF) proposed by Sanderson [11] and Roumeliotis *et al.* [9]. The idea behind EA is simple: Whenever two robots meet, they average their location estimations. EA requires considerably less communication and computation then KF. Furthermore, EA produce better results when no absolute localization information is available to the robots.

In case the robots have no access to absolute localization information, EA's localization error is composed of two components. A constant component and a monotonically increasing time dependent component. The constant component result from the time the error require to propagate between the robots and is a function of the odometry quality and the frequency of meetings. The time dependent component results from the error accumulated by the robots and is a function of the odometry quality and the number of robots i.e. it is independent of the frequency of meetings. In case some robots have access to absolute localization (e.g. a landmark), the localization errors of all robots become bounded.

Simulations showed that the analysis is accurate on the torus. When the environment includes obstacles, the time dependent component is predicted well but the constant is not. This is because p_r, the probability of meeting, was

derived for the torus. When the environment comprises obstacles, the effective p_r is lower then calculated hence the constant component is larger.

In this paper we have used a soft "thermodynamic" analysis i.e. relaxing assumptions were used. The analysis goal was not to derive hard lower or upper bounds but rather to characterize the robots expected behavior. In particular, to predict the expected localization error. We believe that this kind of soft analysis can be beneficial in many other cases.

Acknowledgments. This research was supported by the Technion Goldstein UAV and Satellite Center and by the European Community's FP7-FET program, SMALL project.

References

1. Borenstein, J., Everett, H.R., Feng, L.: Navigating Mobile Robots: Systems and Techniques. A. K. Peters, Ltd., Natick (1996)
2. Elor, Y., Bruckstein, A.M.: A thermodynamic approach to the analysis of multi-robot cooperative localization under independent errors. Tech. rep., Technion (Mar 2010) (under revision for ANTS 2010)
3. Fox, D., Burgard, W., Kruppa, H., Thrun, S.: A probabilistic approach to collaborative multi-robot localization. Autonomous Robots 8(3), 325–344 (2000)
4. Kurazume, R., Nagata, S., Hirose, S.: Cooperative positioning with multiple robots. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, vol. 2 (1994)
5. Martinelli, A.: Improving the precision on multi robot localization by using a series of filters hierarchically distributed. In: Proc. IEEE/RSJ Int. Conf. on Intel. Robots and Systems, San Diego, CA, USA (October 2007)
6. Mourikis, A., Roumeliotis, S.: Optimal sensing strategies for mobile robot formations: Resource-constrained localization. In: Robotics: Science and Systems, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA (June 2005)
7. Mourikis, A., Roumeliotis, S.: Performance analysis of multirobot cooperative localization. IEEE Trans. on Robotics 22(4), 666–681 (2006)
8. Rekleitis, I., Dudek, G., Milios, E.: Multi-robot collaboration for robust exploration. Annals of Math and Artificial Intel. 31(1), 7–40 (2001)
9. Roumeliotis, S., Bekey, G.: Distributed multirobot localization. IEEE Trans. on Robotics and Automation 18(5), 781–795 (2002)
10. Roumeliotis, S.I., Rekleitis, I.M.: Propagation of uncertainty in cooperative multi-robot localization: Analysis and experimental results. Auton. Robots 17(1) (2004)
11. Sanderson., A.C.: A distributed algorithm for cooperative navigation among multiple mobile robots. Advanced Robotics 12, 335–349 (1997)
12. Thrun, S.: Robotic mapping: a survey. In: Exploring Artificial Intel. in the New Millenium, pp. 1–35. Morgan Kaufmann Publishers Inc., San Francisco (2003)
13. Thrun, S., Fox, D., Burgard, W., Dellaert, F.: Robust monte carlo localization for mobile robots. Artificial Intelligence 128(1-2), 99–141 (2001)

An Alternative $ACO_{\mathbb{R}}$ Algorithm for Continuous Optimization Problems

Guillermo Leguizamón[1,⋆] and Carlos A. Coello Coello[2,⋆⋆]

[1] UMI LAFMIA 3175 CNRS, CINVESTAV-IPN
Departmento de Computación, México D.F., México
legui@unsl.edu.ar
[2] CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación, México D.F., México
ccoello@cs.cinvestav.mx

Abstract. The Ant Colony Optimization (ACO) metaheuristic embodies a large set of algorithms which have been successfully applied to a wide range of optimization problems. Although ACO practitioners have a long tradition in solving combinatorial optimization problems, many other researchers have recently developed a variety of ACO algorithms for dealing with continuous optimization problems. One of these algorithms is the so-called $ACO_{\mathbb{R}}$, which is one of the most relevant ACO algorithms currently available for continuous optimization problems. Although $ACO_{\mathbb{R}}$ has been found to be successful, to the authors' best knowledge its use in high-dimensionality problems (i.e., with many decision variables) has not been documented yet. Such problems are important, because they tend to appear in real-world applications and because in them, diversity loss becomes a critical issue. In this paper, we propose an alternative $ACO_{\mathbb{R}}$ algorithm ($DACO_{\mathbb{R}}$) which could be more appropriate for large scale unconstrained continuous optimization problems. We report the results of an experimental study by considering a recently proposed test suite. In addition, the parameters setting of the algorithms involved in the experimental study are tuned using an *ad hoc* tool. Our results indicate that our proposed $DACO_{\mathbb{R}}$ is able to improve both, the quality of the results and the computational time required to achieve them.

1 Introduction

Several extensions of the Ant Colony Optimization (ACO) metaheuristic [4,5] for solving continuous problems currently exist. The first ACO extension designed to operate on continuous search spaces was introduced by Bilchev et al. [3]. After that, several others were introduced (see [13,11,6,7,12,14,9,8]). The version adopted for our study is the one originally proposed by Socha [15]) and further extended by Socha & Dorigo [16].

⋆ On leave of absence from LIDIC - Universidad Nacional de San Luis, San Luis, Argentina.

⋆⋆ The second author is also affiliated to the UMI LAFMIA 3175 CNRS at CINVESTAV-IPN.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 48–59, 2010.

In this work, we propose an alternative $ACO_{\mathbb{R}}$ algorithm for solving large dimensional continuous optimization problems. This study has a recent antecedent (see [10]) in which it was detected that $ACO_{\mathbb{R}}$ [16] had some limitations when dealing with large dimensional problems. Its main problem was a quick loss of diversity, which had a clear negative impact on the quality of the results achieved by the algorithm. In order to deal with such problem, a simple diversity maintenance mechanism was introduced. Here, we propose a mechanism different to the one adopted in [10], which aims to avoid the loss of diversity by using an alternative mechanism to select the kernels that produce new samples on the search space.

Our experimental study includes a recently proposed test suite of continuous optimization problems which are useful to assess the capacity of an algorithm to deal with large dimensional problems. For determining the parameters setting to be used in the experimental study, we used an automatic tool that approximates a prediction model based on a set of observations (outputs of the real algorithm) for specific design points. Such a proposal intends to be the first step towards improving the ACO metaheuristic in order to achieve a design of a more advanced ACO algorithm which is competitive with respect to other state-of-the-art metaheuristic algorithms used for continuous optimization problems.

The remainder of this paper is organized in the following way: Section 2 briefly describes the original version of the $ACO_{\mathbb{R}}$ algorithm and Section 3 presents the alternative $ACO_{\mathbb{R}}$ algorithm (called $DACO_{\mathbb{R}}$). The section about the experimental study (Section 4) involves three important subsections: Section 4.1 describes the set of test problems adopted; Section 4.2 presents the results of a preliminary study of $ACO_{\mathbb{R}}$ and $DACO_{\mathbb{R}}$ on the test suite by using an *ad hoc* tool to tune some selected parameters of the algorithms, and Section 4.3 shows a comparative analysis of the obtained results from algorithms $ACO_{\mathbb{R}}$ and $DACO_{\mathbb{R}}$. Finally, in Section 5 we discuss the main achievements of the experimental study. In addition, some lines of future research are also considered.

2 The $ACO_{\mathbb{R}}$ Algorithm

The $ACO_{\mathbb{R}}$ algorithm was designed with the aim of obtaining a set of *probability density functions* (PDFs). Each PDF is obtained from the search experience and is used to incrementally build a solution $\mathbf{x} \in \mathbb{R}^n$ considering in turn each component x_j ($\forall j = 1 \ldots n$). To approximate a multimodal PDF, Socha & Dorigo [16] proposed a Gaussian kernel which is defined as a weighted sum of several one-dimensional Gaussian functions $g_{ij}(x)$ as follows:

$$G^j(x) = \sum_{i=1}^{k} \omega_i g_{ij}(x) = \sum_{i=1}^{k} \omega_i \frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{-\frac{(x-\mu_{ij})^2}{2(\sigma_{ij})^2}} \tag{1}$$

where $j \in \{1, \ldots, n\}$ identifies the number of dimension, i.e., $ACO_{\mathbb{R}}$ uses as many Gaussian kernel PDFs as the number of dimensions of the problem. In addition, G^j is parameterized with three vectors: $\boldsymbol{\omega}$, the vector of weights associated with

the individual Gaussian functions; $\boldsymbol{\mu}_j$, the vector of means; and $\boldsymbol{\sigma}_j$, the vector of standard deviations. All these vectors have cardinality k, which constitutes the number of Gaussian functions involved.

In ACO$_{\mathbb{R}}$, a solution archive called T is used to keep track of a number of solutions. The cardinality of archive T is k, that is, the number of kernels that conform the Gaussian kernel. For each solution $\mathbf{x}_i \in \mathbb{R}^n$, ACO$_{\mathbb{R}}$ maintains the corresponding values of each problem dimension, i.e., $x_{i1}, \ldots, x_{in}$, and the value of the objective function $f(\mathbf{x}_i)$ which are stored satisfying that $f(\mathbf{x}_1) \leq \ldots \leq f(\mathbf{x}_i) \leq \ldots f(\mathbf{x}_k)$. On the other hand, the vector of weights $\boldsymbol{\omega}$ should satisfy that $\omega_1 \geq \ldots \geq \omega_l \geq \ldots \geq \omega_k$. The solutions in T are, therefore, used to dynamically generate probability density functions involved in the Gaussian kernels. More specifically, in order to obtain the Gaussian kernel G^j, the three parameters $\boldsymbol{\omega}$, $\boldsymbol{\mu}_j$, and $\boldsymbol{\sigma}_j$ need to be calculated. Thus, for each G^j, the values of the j-th variable of the k solutions in T become part of the elements of vector $\boldsymbol{\mu}_j$, that is, $\boldsymbol{\mu}_j = (\mu_{1j}, \ldots, \mu_{kj}) = (x_{1j}, \ldots, x_{kj})$. On the other hand, each component of the deviation vector $\boldsymbol{\sigma}_j = (\sigma_{1j}, \ldots, \sigma_{kj})$ is obtained as:

$$\sigma_{ij} = \xi \sum_{e=1}^{k} \frac{|x_{ej} - x_{ij}|}{k-1} \tag{2}$$

where $i \in \{1, \ldots, k\}$ is the kernel number with respect to which the deviation is calculated and $\xi > 0$ which is the same for all dimensions, has an effect similar to that of the pheromone evaporation rate in ACO. Thus, the higher the value of ξ, the lower the convergence speed of the algorithm.

The pheromone update is achieved by considering a set A[1] of size N_a which maintains the newly generated solutions regarding equation (1). The new T (in the next algorithm iteration) is obtained as $T(t+1) = FIRST_k(Rank(T(t) \oplus A(t)))$, i.e., the old solutions in the archive T plus the set of newly created solutions A are ranked and then, the first k best k solutions are selected. In other words, the old solutions compete against the newly generated ones to conform the updated T which maintains its cardinality (k) through the whole search process.

3 The Proposed Alternative ACO$_{\mathbb{R}}$ Algorithm (DACO$_{\mathbb{R}}$)

The proposed alternative ACO$_{\mathbb{R}}$ algorithm is straightforward. The main objective is to keep the diversity as long as possible in order to explore more regions of the search space before converging to a possible local optimum from which is usually impossible to escape unless some mechanism is implemented to improve the diversity in the population. Our proposed algorithm, called DACO$_{\mathbb{R}}$ ('D' stands for Diversity) is designed following the same basic principle of ACO$_{\mathbb{R}}$, i.e., from a set of k kernels, a new set of Na solutions is generated via the multimodal kernels (see equation (1)). However, DACO$_{\mathbb{R}}$ always generates $Na = k$ new solutions by considering an alternative approach to select the kernels. More

[1] Set A represents the set of ants according to Socha & Dorigo [16].

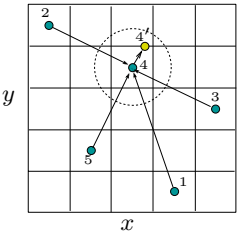


Fig. 1. A possible LHS distribution for $k = 5$ on a hypothetical search space of dimension $n = 2$. The center point in the circle represents the current kernel (number 4) from which a new point is generated by considering the remaining $5 - 1$ kernels.

precisely, DACO$_{\mathbb{R}}$ starts from an initial population of k kernels distributed evenly on the whole problem search space. In our case, we have adopted Latin Hypercube Sampling (LHS) under which the search space is divided into k intervals. Figure 1 shows a possible LHS distribution of $k = 5$ kernels on a hypothetical search space of dimension $n = 2$.

To generate the new set of solutions $A(t)$ from the actual set of kernels $T(t)$, DACO$_{\mathbb{R}}$ considers two different ways of selecting the kernel from $T(t)$ to produce the corresponding solution in $A(t)$. The first one is as follows: when generating solution i in $A(t)$, the selected kernel from $T(t)$ is the number i, i.e., the solution generated at position i in $A(t)$ will be obtained through a Gaussian distribution with $\boldsymbol{\mu} = \boldsymbol{x}_i$ and a deviation $\boldsymbol{\sigma}$ determined by the remaining set $\{1, \ldots, i-1, i+1, \ldots, k\}$ of $k-1$ kernels in $T(t)$. The newer solution is included in $A(t)$ only if its corresponding objective value is improved with respect to kernel i in $T(t)$; otherwise, the old kernel is copied to $A(t)$ as the new solution generated. In this way, the algorithm behaves as a local explorer around each kernel.

The second approach to generate a solution is by considering the current best kernel in $T(t)$ to generate a particular solution in $A(t)$. Thus, the algorithm globally exploits the best solution in the current population $T(t)$, i.e., the solution generated at position i in $A(t)$ will be obtained through a Gaussian distribution with $\boldsymbol{\mu} = \boldsymbol{x}_{i_{best}}$ and a deviation $\boldsymbol{\sigma}$ determined by the remaining set $\{1, \ldots, i_{best}-1, i_{best}+1, \ldots, k\}$ of $k-1$ kernels in $T(t)$. The newer solution is included in $A(t)$ only if its corresponding objective value is improved with respect to kernel i in $T(t)$; otherwise, the old kernel is copied to $A(t)$ as the new solution generated.

To choice between the two ways of constructing $A(t)$ is determined by a parameter q as expressed in equation (3). It should be noticed that parameter q in DACO$_{\mathbb{R}}$ is different from parameter q in ACO$_{\mathbb{R}}$. In our DACO$_{\mathbb{R}}$ algorithm, q determines the way of obtaining some element in $A(t)$ whereas in ACO$_{\mathbb{R}}$, this parameter determines the relative weight of the ranked kernels. In addition, it also important to note that our DACO$_{\mathbb{R}}$ algorithm does not need to sort, at each iteration, the set of kernels as required in ACO$_{\mathbb{R}}$. Parameter ξ is used in DACO$_{\mathbb{R}}$ in the same way as in ACO$_{\mathbb{R}}$.

Algorithm 1. Outline of DACO$_{\mathbb{R}}$ algorithm

```
1: Init_LHS(T);
2: Get_s( σ );
3: for t ∈ 1 : t_max do
4:    A = BuildSolsNew(T, σ);
5:    T = Sel_Best_one_to_one(T ,A);
6:    Get_s(σ);
7: end for
```

$$A_{ij} = \begin{cases} \text{gen\_xj}(T_{ij},\, \sigma_j^i), & q > rand(0,1) \text{ (exploration)}; \\ \text{gen\_xj}(T_{i_b j},\, \sigma_j^{i_b}), & \text{otherwise (exploitation)}. \end{cases}$$
$$\text{where } i = 1, \ldots, N_a \text{ (recall that } N_a = k) \text{ and } j = 1, \ldots, n, \quad \text{and} \quad i_b \text{ is the index to the best current solution in } T(t). \tag{3}$$

Algorithm 1 outlines the main components of DACO$_{\mathbb{R}}$. Init_LHS() gives the initial set of k kernels through LHS; BuildSolsNew() is in charge of generating $A(t)$ by following the procedure explained before, i.e., either by using a local or a global mechanism. Sel_Best_one_to_one() selects in a one-to-one correspondence the best solutions between $A(t)$ and $T(t)$; and Get_s() obtains the new deviation vectors according to the new populations of kernels recently generated $T(t+1)$.

4 Experimental Study

In this section, we present the experimental study that includes: a) a short description of the test suite adopted to assess the performance of the ACO$_{\mathbb{R}}$ and DACO$_{\mathbb{R}}$ algorithms; b) a preliminary study to determine an appropriate parameters setting of the algorithms (for that sake, we have used an automatic tool proposed by Bartz-Beielstein [1] called SPOT [2] (Sequential Parameter Optimization Tool); and c) a comparison between ACO$_{\mathbb{R}}$ and DACO$_{\mathbb{R}}$ for the adopted test suite by considering $n \in \{30, 50, 100, 200, 500\}$ dimensions for each of the six problems. All the experiments, except for those corresponding to the preliminary study, were run on a PC having an Intel Pentium (R) 4 processor, running at 3.00Gz, and with 1Gb of RAM. The ACO$_{\mathbb{R}}$ and DACO$_{\mathbb{R}}$ algorithms were implemented in the C programming language.

4.1 The Adopted Test Suite

We selected 6 problems from the benchmark functions prepared for the "Special Session and Competition on Large Scale Global Optimization" at the *2008 IEEE Congress on Evolutionary Computation (CEC'08)* [17]. The problems represent a set of scalable functions for high-dimensional optimization. See Table 1 for a description of these problems and their corresponding optimum values. Particularly, the objective of this special session was to bring to the research community

Table 1. Test suite proposed by Tang et al. [17]

Benchmark Problems	Search Range	$f(\mathbf{x}^*)$
$f_1(\mathbf{x}) = \sum_{j=1}^{n} z_j + f\_bias_1, \mathbf{z} = \mathbf{x} - \mathbf{o}$ $\mathbf{o} = (o_1, o_2, \ldots, o_n)$; the shifted global optimum	[-100,100]	-450
$f_2(\mathbf{x}) = max_j\{\lvert z_j\rvert, 1 \leq j \leq n\} + f\_bias_2, \mathbf{z} = \mathbf{x} - \mathbf{o}$ $\mathbf{o} = (o_1, o_2, \ldots, o_n)$; the shifted global optimum	[-100,100]	-450
$f_3(\mathbf{x}) = \sum_{j=1}^{n-1}(100 \cdot (z_j^2 - z_j)^2 + (z_j - 1)^2) + f\_bias_3$, $\mathbf{z} = \mathbf{x} - \mathbf{o} + \mathbf{1}$; $\mathbf{o} = (o_1, o_2, \ldots, o_n)$; the shifted global optimum	[-100,100]	390
$f_4(\mathbf{x}) = \sum_{j=1}^{n}(z_j^2 - 10 \cdot \cos(2\pi z_j) + 10) + f\_bias_4, \mathbf{z} = \mathbf{x} - \mathbf{o}$ $\mathbf{o} = (o_1, o_2, \ldots, o_n)$; the shifted global optimum	[-5,5]	-330
$f_5(\mathbf{x}) = \sum_{j=1}^{n} \frac{z_j^2}{4000} - \prod_{j=1}^{n} \cos(\frac{z_j}{\sqrt{j}}) + 1 + f\_bias_5, \mathbf{z} = \mathbf{x} - \mathbf{o}$ $\mathbf{o} = (o_1, o_2, \ldots, o_n)$; the shifted global optimum	[-600,600]	-180
$f_6(\mathbf{x}) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n} z_j^2})$ $-\exp(\frac{1}{n}\sum_{j=1}^{n}\cos(2\pi z_j)) + 20 + f\_bias_6)$, $\mathbf{z} = \mathbf{x} - \mathbf{o}$; $\mathbf{o} = (o_1, o_2, \ldots, o_n)$; the shifted global optimum	[-32,32]	-140

newer and more challenging problems to assess current nature-inspired optimization algorithms as well as other, novel optimization algorithms.

4.2 Parameters Settings for ACO$_{\mathbb{R}}$ and DACO$_{\mathbb{R}}$

In order to conduct the preliminary study and establish the most appropriate parameters setting for our proposed approach, it was necessary the integration of the algorithms ACO$_{\mathbb{R}}$ and DACO$_{\mathbb{R}}$ (implemented in C) with SPOT (implemented in MATLAB) through the compiler MEX. After tuning the corresponding parameters, all the algorithms ran as standalone processes in the usual way.

As an optimization algorithm, SPOT includes several specific parameters that must be provided when applied to a particular algorithm. In our case, we used the default parameters setting for this tool (e.g., the sampling procedure applied is the Latin Hypercube Sampling where the number of design points is set to 16 by default). Additionally, SPOT needs to run the algorithm (either ACO$_{\mathbb{R}}$ or DACO$_{\mathbb{R}}$) to fit a model based on a sample of observations; accordingly, some fixed parameters (not included in the algorithm's design, explained below) of the respective algorithm under study need to be provided. For example: the problem dimension ($n = 100$ was the chosen setting), the maximum number of iterations (was set to $t_{max} = 1000$), and the number of kernels and ants (they were set respectively to $k = 50$ and $N_a = 50$). It should be noticed that for DACO$_{\mathbb{R}}$, Na is always set as k (i.e., $Na = k$). In addition, the setting for the number of dimensions and maximum number of iterations, was only used to calibrate the selected parameters (q and ξ) as explained next. For the comparative study (see Section 4.3) the respective settings for these parameters are different (except for k and N_a).

The definition of the *Problem Design* (X_P regarding the terminology taken from [1]) for both algorithms includes function F which is expressed as:

$$F(\mathbf{x}, n) = \sum_{i=1}^{6} (f_i(\mathbf{x}, n) - f_i^*)/f_i^* \tag{4}$$

where f_i represents one of the six functions from the test suite studied (presented in Table 1), n is the problem's dimensionality (all these functions are scalable), and f_i^* represents the optimal value for function i (it must be noticed that for any dimension n, the optimal values remain the same). Thus, F expresses the summation of the percentage error over the six functions. In this way, we apply SPOT as considering only one problem in the process of tuning the corresponding algorithms' parameters.

We first define the *Algorithm Design* for $\text{DACO}_{\mathbb{R}}$ (see [1]) as $X_{\text{DACO}_{\mathbb{R}}}$, the region determined by parameters q and ξ as follows: $0 \leq q \leq 1$ and $0 \leq \xi \leq 1$. After the initial application of SPOT to calibrate these parameters, we observed that the regions determined by $q \in (0.5, 1]$ and $\xi \in [0, 0.5)$ can be eliminated from the experimental study due to the poor performance of the algorithm for such parameter values. Accordingly, we redefined $X_{\text{DACO}_{\mathbb{R}}}$ as the region determined by $0 \leq q \leq 0.5$ and $0.5 \leq \xi \leq 1$. The best parameters setting for $\text{DACO}_{\mathbb{R}}$ was: $q = 0.1172$ and $\xi = 0.6063$. Figure 2 shows the output from SPOT for the regions considered of the algorithm's design ($X_{\text{DACO}_{\mathbb{R}}}$) with respect to parameters q and ξ. On the left, we can observe the response surface of the predicted values (PV) of function F for $\text{DACO}_{\mathbb{R}}$. On the right, we show the corresponding surface of the Mean Square Error (MSE).

Similarly, for $\text{ACO}_{\mathbb{R}}$, $X_{\text{ACO}_{\mathbb{R}}}$ was determined by considering the region $0 \leq q \leq 0.5$ and $0.5 \leq \xi \leq 1$ from which SPOT reported $q = 0.0103$ and $\xi = 0.8257$ as the best corresponding parameters for the $\text{ACO}_{\mathbb{R}}$ algorithm.

Finally, it must be remarked that for both algorithms ($\text{ACO}_{\mathbb{R}}$ and $\text{DACO}_{\mathbb{R}}$), the initial population of kernels was obtained from input files previously generated by using the MATLAB function `lhsdesign`.

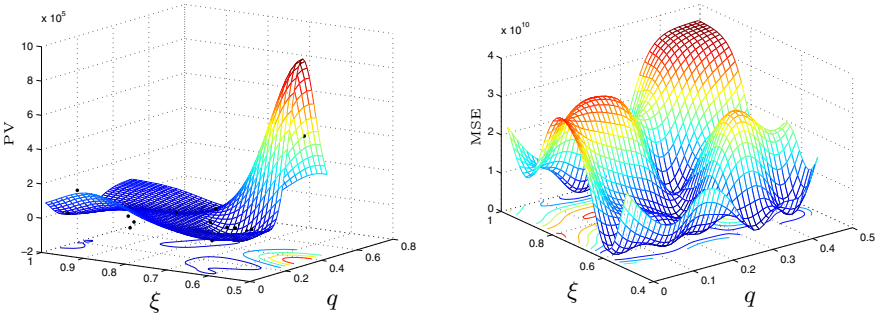
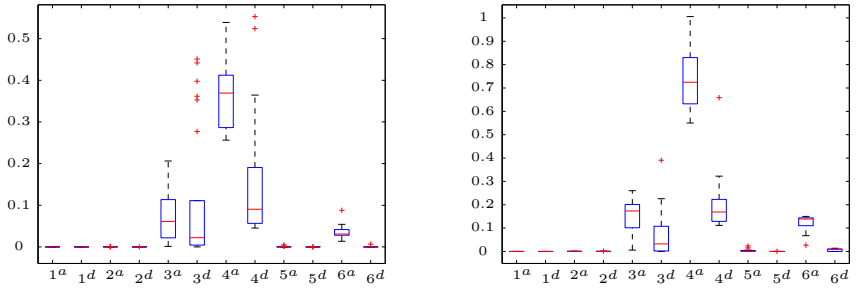


Fig. 2. Regions of the algorithm's design ($X_{\text{DACO}_{\mathbb{R}}}$) with respect to parameters q and ξ. (Left)- response surface of the predicted values (PV) of function F for $\text{DACO}_{\mathbb{R}}$ and (Right)- the corresponding surface of the Mean Square Error (MSE).

4.3 Performance of $ACO_{\mathbb{R}}$ and $DACO_{\mathbb{R}}$ on the Selected Problems

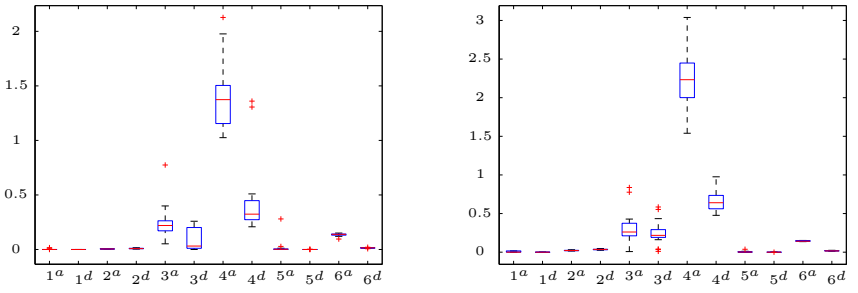
This section presents a comparative study of $ACO_{\mathbb{R}}$ and $DACO_{\mathbb{R}}$ on the six problems presented in Section 4.1. The parameters settings for these algorithms are as follows. For both algorithms, $k = Na = 50$, t_{max} was set respectively to 6000, 10000, 14000, 20000, 40000, and 100000 when running the algorithms with problems of dimensions $n = 30$, $n = 50$, $n = 70$, $n = 100$, $n = 200$, and $n = 500$. These values for t_{max} were selected considering the criteria followed in [17]. Particularly, we considered twice the maximum number of function evaluations (FES) proposed for dimension $n = 100$. Accordingly, we obtained the corresponding t_{max} values proportionally for each dimension studied. Under these parameters settings, both algorithms ran for the same number of function evaluations for each of the problems and dimensions considered. With respect to the remaining parameters (i.e., q and ξ), the used values were those reported by SPOT which are, for $DACO_{\mathbb{R}}$, $q = 0.1172$ and $\xi = 0.6063$; whereas for $ACO_{\mathbb{R}}$, are $q = 0.0103$ and $\xi = 0.8257$. Both algorithms were run considering 25 random seeds for each combination of problem, dimension, and parameters setting.

The results are shown in all the figures displayed in the Appendix. We adopted boxplots to show the distribution of the results expressed as the percentage of the error with respect to the optimum values. We divided the presentation of the results based on the problem's dimension. On the one hand, Figures 3 and 4 show respectively the results for dimensions $n = 30, 50$ and $n = 70, 100$. On the other hand, and because of the large differences found in the percentage error for the larger dimensions considered in this work ($n = 200$ and $n = 500$), we split the presentation in three figures for: i) problems 1,2,4, and 5 with $n = 200, 500$ (Figure 5); ii) problems 3 and 6 with $n = 200$ (Figure 6); and iii) problems 3 and 6 with $n = 500$ (Figure 7). The x-axis in each figure indicates the problem number whereas the corresponding super-index represents the corresponding applied algorithm (a stands for $ACO_{\mathbb{R}}$ and d stands for $DACO_{\mathbb{R}}$). The non-parametric Mann-Whitney-Wilcoxon test at a level of 5% of confidence was applied to assess



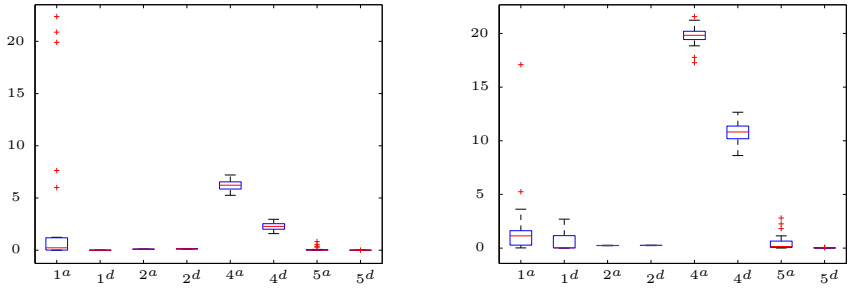
(a) Dimension $n = 30$ (b) Dimension $n = 50$

Fig. 3. Percentage error for the six benchmark problems with dimension $n \in \{30, 50\}$



(a) Dimension $n = 70$ (b) Dimension $n = 100$

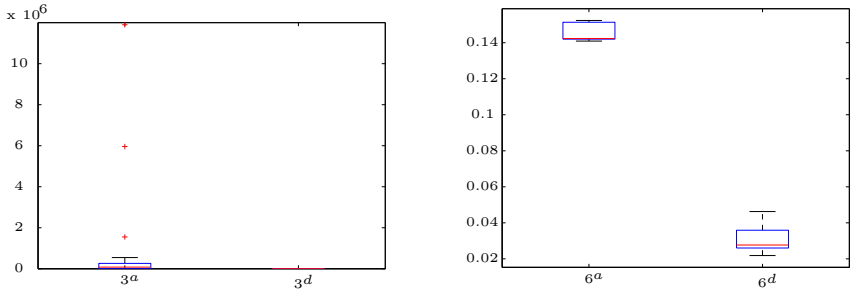
Fig. 4. Percentage error for the six benchmark problems with dimension $n \in \{70, 100\}$



(a) Dimension $n = 200$ (b) Dimension $n = 500$

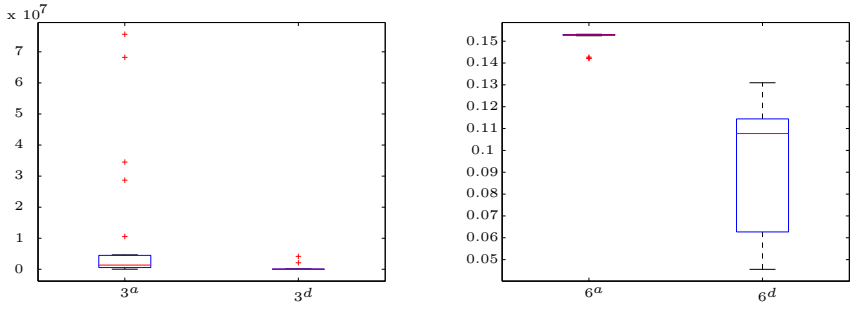
Fig. 5. Percentage error for problems 1, 2, 4, and 5 with dimension $n \in \{200, 500\}$

the significance on the differences on the corresponding medians of $\text{DACO}_{\mathbb{R}}$ with respect to $\text{ACO}_{\mathbb{R}}$. Thus, a p-value < 0.05 indicates that based on the median values $\text{DACO}_{\mathbb{R}}$ outperforms $\text{ACO}_{\mathbb{R}}$. It is worth mentioning that the statistical test was applied considering, for each dimension, a sample of 25×6 points (i.e., all the percentage error values for each problem and run were collected as one sample). The p-values are respectively $4.0459e-007$, $3.7094e-0051$, 0.0083, 0.0117, and $2.6185e-005$ for dimensions 30, 50, 70, 100, 200, and 500. From this statistical point of view, $\text{DACO}_{\mathbb{R}}$ outperforms $\text{ACO}_{\mathbb{R}}$ for all problems and dimensions considered. More precisely, when taking into account the shape and location of the boxplots for all dimensions and considering one problem in turn, we can observe a similar behavior of both algorithms. On the one hand, the algorithms scale fairly well with larger dimensions for problems 1, 2, and 5. Also for these problems, it can be seen that both algorithms preformed robustly and achieved high quality results (mainly for dimensions $n \in \{30, 50, 70, 100\}$). However, $\text{DACO}_{\mathbb{R}}$ found the best results for these problems with all the dimensions



(a) Dimension $n = 200$, problem 3 (b) Dimension $n = 200$, problem 6

Fig. 6. Percentage error for problems 3 and 6 with dimension $n = 200$



(a) Dimension $n = 500$, problem 3 (b) Dimension $n = 500$, problem 6

Fig. 7. Percentage error for problems 3 and 6 with dimension $n = 500$

tested. When increasing the problems' dimensionality for problems 1,2, and 5 (i.e., $n = 200$ and $n = 500$) we found a less robust behavior and results of lower quality for algorithm ACO$_{\mathbb{R}}$. On the other hand, problems 3, 4, and 6 represent a challenge for both algorithms. It can be clearly observed that there was a large increase in the percentage error as the problem dimensionality increased. Although both algorithms have difficulties to solve and scale on these three problems, the behavior of DACO$_{\mathbb{R}}$ is superior to that of ACO$_{\mathbb{R}}$. Finally, is it worth remarking that DACO$_{\mathbb{R}}$ needed less CPU time to complete the same number of function evaluations than the ACO$_{\mathbb{R}}$ algorithm. Indeed, our proposed approach required about 25% less CPU time, on average, than ACO$_{\mathbb{R}}$, for all the problems and dimensions tested in our study. This can be explained based on the fact that our proposed DACO$_{\mathbb{R}}$ algorithm does not include the sorting procedure used by the original ACO$_{\mathbb{R}}$ to produce, at each iteration, a ranked set of kernels. In DACO$_{\mathbb{R}}$, it is only necessary to maintain an index to the current best kernel (i_{best} in equation (3)).

5 Discussion and Conclusions

In this work we presented $DACO_{\mathbb{R}}$, an alternative to the $ACO_{\mathbb{R}}$ algorithm for dealing with large dimensional continuous problems. The achieved results show the potential of our proposed $DACO_{\mathbb{R}}$ algorithm to solve large scale optimization problems. Our results lead us to think about other possible modifications, including more sophisticated mechanisms to control the intensification and the diversification during the search. This could strengthen the position of the ACO metaheuristic with respect to state-of-the-art algorithms in current use for large dimensional continuous optimization problems (e.g., differential evolution and evolution strategies). Our future work will include the incorporation of a mechanism to better control the region of local exploration as well as to automatically decide between a global and a local exploration based on the diversity observed in the current population of kernels. Improved versions of $DACO_{\mathbb{R}}$ should certainly be compared with some state-of-the-art metaheuristics for continuous optimization problems. Additionally, it is important to acquire more experience in the use of tools (either SPOT or some other approach) to appropriately set the main parameters of the future version of the algorithms to be studied.

Acknowledgements. The first author acknowledges the support from the UMI-LAFMIA 3175 CNRS at CINVESTAV-IPN and from the Universidad Nacional de San Luis, Argentina. The second author gratefully acknowledges support from CONACyT project no 103570.

References

1. Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation: The New Experimentalism. Natural Computing Series. Springer, New York (2006)
2. Bartz-Beielstein, T., Preuss, M.: Spot, sequential parameter optimization tool, http://www.gm.fh-koeln.de/campus/personen/lehrende/thomas.bartz-beielstein/00489/
3. Bilchev, G., Parmee, I.: The Ant Colony Metaphor for Searching Continuous Design Spaces. In: Fogarty, T.C. (ed.) Evolutionary Computing. AISB Workshop, pp. 25–39. Springer, Sheffield (April 1995)
4. Corne, D., Dorigo, M., Glover, F. (eds.): New Ideas in Optimization. McGraw-Hill International, London (1999)
5. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
6. Dréo, J., Siarry, P.: A New Ant Colony Algorithm Using the Heterarchical Concept Aimed at Optimization of Multiminima Continuous Functions. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) Ant Algorithms 2002. LNCS, vol. 2463, pp. 216–221. Springer, Heidelberg (2002)
7. Dréo, J., Siarry, P.: Continuous Interacting Ant Colony Algorithm Based on Dense Heterarchy. Future Generation Comp. Syst. 20(5), 841–856 (2004)
8. Hu, X., Zhang, J., Li, Y.: Orthogonal methods based ant colony search for solving continuous optimization problems. J. Comput. Sci. Technol. 23(1), 2–18 (2008)
9. Kong, M., Tian, P.: A direct application of ant colony optimization to function optimization problem in continuous domain. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 324–331. Springer, Heidelberg (2006)

10. Leguizamón, G., Coello Coello, C.A.: A Study of the Scalability of $ACO_{\mathbb{R}}$ for Continuous Optimization Problems. Tech. Rep. EVOCINV-01-2010, Evolutionary Computation Group at CINVESTAV, Departamento de Computación, CINVESTAV-IPN, México (February 2010)
11. Ling, C., Jie, S., Ling, Q., Hongjian, C.: A Method for Solving Optimization Problems in Continuous Space Using Ant Colony Algorithm. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) ANTS 2002. LNCS, vol. 2463, pp. 288–289. Springer, Heidelberg (2002)
12. Ling Chen, J., Shen, L.Q., Chen, H.: An improved ant colony algorithm in continuous optimization. Journal of Systems Science and Systems Engineering 12(2), 224–235 (2003)
13. Monmarché, N., Venturini, G., Slimane, M.: On how pachycondyla apicalis ants suggest a new search algoritm. Future Generation Computer Systems 16, 937–946 (2000)
14. Pourtakdoust, S., Nobahari, H.: An Extension of Ant Colony Systems to Continuos Optimization Problems. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 294–301. Springer, Heidelberg (2004)
15. Socha, K.: ACO for continuos and mixed-variable optimization. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 25–36. Springer, Heidelberg (2004)
16. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. European Journal of Operational Research 185(3), 1155–1173 (2008)
17. Tang, K., Yao, X., Suganthan, P.N., MacNish, C., Chen, Y.P., Chen, C.M., Yang, Z.: Benchmark Functions for the CEC 2008 Special Session and Competition on Large Scale Global Optimization. Tech. rep., Nature Inspired Computation and Applications Laboratory, USTC, China (2007)

An Efficient Optimization Method for Revealing Local Optima of Projection Pursuit Indices

Souad Larabi Marie-Sainte[1], Alain Berro[1], and Anne Ruiz-Gazen[2]

[1] IRIT-UT1, UMR 5505, CNRS
Université Toulouse 1 - Capitole, Toulouse, France
{souad.larabi,alain.berro}@irit.fr
[2] Toulouse School of Economics (GREMAQ)
Université Toulouse 1 - Capitole, Toulouse, France
ruiz@cict.fr

Abstract. In order to summarize and represent graphically multidimensional data in statistics, projection pursuit methods look for projection axes which reveal structures, such as possible groups or outliers, by optimizing a function called projection index. To determine these possible interesting structures, it is necessary to choose an optimization method capable to find not only the global optimum of the projection index but also the local optima susceptible to reveal these structures. For this purpose, we suggest a metaheuristic which does not ask for many parameters to settle and which provokes premature convergence to local optima. This method called Tribes is a hybrid Particle Swarm Optimization method (PSO) based on a stochastic optimization technique developed in [2]. The computation is fast even for big volumes of data so that the use of the method in the field of projection pursuit fulfills the statistician expectations.

1 Introduction

Exploratory projection pursuit techniques aim to reveal visually an interesting structure hidden within multivariate data ([9], [10], [1]). This family of statistical methods consists in detecting interesting linear projections by optimizing a predetermined function called projection index that measures in some sense the "interestingness" of a projection.

The projection pursuit is based on two important elements: the projection index and the optimization algorithm. The literature exposes several projection indices and optimization methods. These methods are global optimization methods such as the gradient's method [10], the ascent's method ([7], [13], [14], [15]), the quasi-Newton's method ([7], [15]) and some modified version of Newton's method [14]. Several projections of the data may reveal interesting structures. So, in order to obtain different local optima, the aforementioned algorithms work in the following way. They look for a global optimum of the projection index and when a solution is found, it is removed from the space of solutions for instance by projecting the data in the orthogonal space of the global solution. Then, the index is optimized again in order to find other solutions. Several projections (local

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 60–71, 2010.

optima) in the initial search space may not be detected when we consider the successive orthogonal spaces. Furthermore, the optimization methods quoted above require the calculation of the gradient and may ask for a meticulous choice of an initial point (initial solution). Our objective is not only to find the global optimum of the index but also the local optima to reveal these possible interesting structures. We also wish to suggest to the statistician an algorithm without parameter tuning and which enough explores the space to find various local optima of the index without considering orthogonal spaces. Furthermore, the problem of the optimization of these projection indices is complex and expensive in computing time and the solutions proposed to maximize (minimize) these indices are not always convincing. Therefore, the projections pursuit methods are little used and absent from the well known statistical softwares (except Matlab [12] or quasi-clones, like SciLab and Octave, and GGobi [5]). Our purpose is to propose powerful and fast algorithm allowing the detection of several local optima.

The Particle Swarm optimization (PSO) and Tribes are metaheuristics that appeared recently. They differ from the other evolutionary methods (typically, the genetic algorithms) and are based on the notion of cooperation between agents (particles). The information exchanged between particles gets to resolve difficult problems. These techniques present some interesting peculiarities, among others, the notion of efficiency due to the collaboration rather than the competition. Furthermore, the fact that these methods converge early to local optima is an interesting feature in order to find new potentially interesting projections. In a first work, we used the PSO and the genetic algorithms to optimize certain projection indices and both optimization methods have proven their efficiency but they need parameters to be tuned. Note that PSO is also used for Projection Pursuit in [16] in the context of regression.

Contrary to PSO, Tribes is presented as a black box, because it possesses no parameter to settle and it easily exhibits satisfactory performances. In this technique, particles are divided into several tribes or groups of variable size. Tribes method presents the risk of a too fast convergence, which can be translated by the fact that it finds local optima. To remedy this problem, Clerc ([3], [4]) proposed a new version of Tribes. As far as our objective is not only to find the global optimum but also several potential local optima, we prefer to apply the native version. Tribes was never used in the field of projections pursuit. Its application in this article shows that it can lead to better results than the classical PSO as shown in the previous work.

In this paper, we present a comparison of the Tribes technique with the classical PSO version applied to the exploratory projections pursuit to optimize one-dimensional projection index. We focus on the search of clusters among any other interesting structure such as outliers. In section 2, we introduce briefly the problem of projection pursuit and the two projection indices we focus on. Section 3 presents the technique of particle swarm optimization briefly and the technique of Tribes in more detail. The last section is dedicated to the comparison of these two techniques on some small data sets.

2 Exploratory Projection Pursuit

The Exploratory Projection Pursuit (EPP) techniques consist in the search for hidden aspects within a big volume of data [8]. The objective of these exploratory techniques is to look for low (one, two or three) dimensional projections that provide the most revealing views of the full-dimensional data. The search for such projections requires the definition of a numerical index $I(a)$ for every projection a. The intent of this index is to capture nonlinear structures present in the distribution of the projected data. This function is defined so that the interesting projections correspond to the global optimum and to the local optima of this function.

Principal components analysis (PCA) is a familiar exploratory technique of this kind, it is a projection pursuit method where the index of interestingness represents the variance of the projected data. Its efficiency has been relativized [10] because certain important projections may not appear in the principal subspaces, even if their dimension is small. Furthermore, the maximization of this index (the variance) can be solved by using the spectral decomposition so that PCA does not need any optimization algorithm.

As in many situations of data analysis, we consider N individuals characterized by P variables. To every individual corresponds a vector X_i in $\mathbb{R}^P (i = 1, \cdots, N)$ which is assimilated to a matrix column, the transposed of these vectors leads to a matrix X with dimension $N \times P$. Unfortunately, it is not possible to visualize points in P-dimensional space if P is upper to 3. However, it is possible to project a P-dimensional set of points onto a one-dimensional line. The projection is a linear function of $\mathbb{R}^P$ towards $\mathbb{R}$ of N observations $X_1, \cdots, X_N$ such as $z = Xa$. The P-vector a defines a linear transformation and the N column-vector z corresponds to the projected data coordinates. The problem consists in determining a projection a. As usual in EPP, we suppose that the data are spherical (by transforming the data accordingly), such that the mean vector $E(X_i) = 0$ and the covariance matrix $V(X_i) = I_P$ where I_P denote the identity P-dimensional matrix. By considering spherical data, PP is going beyond the first and second moments of the data which are already taken into account in standard analysis such as Principal Components analysis.

There are many possible projection indices, the present paper focus on a one dimensional polynomial-based index named the Friedman index [7] and a moment-based index called the kurtosis index [14]. We limit ourselves to a brief definition of these two indices.

2.1 The Friedman Index

This index is based on the Legendre polynomials [7]. It measures the departure between the density of the projected data and the normal density which is assumed to correspond to a non-interesting projection. The formula is given as follows:

$$I_h^F(a) = \sum_{j=1}^{h} \frac{2j+1}{2} \left[\frac{1}{N} \sum_{i=1}^{N} L_j\{2\Phi(X_i) - 1\} \right]^2 \quad (1)$$

where Φ is the univariate standard normal distribution. The recursive definition of the Legendre polynomials is given by:

$$\begin{array}{l} L_0(r) = 1, L_1(r) = r, L_2(r) = \frac{1}{2}(3r^2 - 1), \\ L_j(r) = \frac{1}{j}(2j-1)rL_{j-1}(r) - (j-1)L_{j-2}(r) \text{ pour } j \geq 3 \end{array} \quad (2)$$

The choice of the value of h depends on the data dimension P and the sample size N. In the present article h is fixed to 3 according to the recommendations given in [7] and [15].

2.2 The Kurtosis Index

This index is based on the fourth moment of the projected data distribution [14]. It is the kurtosis coefficient of the projected data. The directions are chosen by minimizing and maximizing this coefficient. The minimization of the kurtosis coefficient implies the maximization of the "bimodality" of the projections, that leads to the determination of clusters, whereas its maximization leads to the detection of outliers [14]. The index is defined as follows:

$$I_k(a) = \sum_{i=1}^{N} (a^T X_i)^4 \quad (3)$$

3 Bio-inspired Algorithms

The optimization algorithm is an important choice in the projection pursuit problem. It consists in finding the directions which maximize (or minimize) the projection index I. This section presents the PSO and Tribes which are bio-inspired algorithms. In other words, there are iterative stochastic methods for global optimization which are inspired by the theory of the biological populations evolution. One of the interests to study these approaches is to develop an algorithm with powerful ability to find out the global and the local optima of the optimization problem. These methods develop a set of solutions with the purpose to find the best results.

3.1 Particle Swarm Optimization (PSO)

The PSO algorithm is an optimization metaheuristic method, invented by Eberhart and Kennedy in 1995 [11]. This method incorporates concepts that lead particles to converge gradually to a local optimum. The PSO algorithm is initialized with a swarm of random candidate solutions, called particles. All the particles have fitness values which are evaluated by the fitness function to be optimized, and are assigned a randomized velocity at the beginning of optimization and are iteratively moved through the problem's searching space. Each particle tries to improve its performance according to its own experience and the experience of its environment.

If $X_m(t)$ represents the position of the particle m to the iteration t, then its velocity at iteration $t+1$ is defined by:

$$V_m(t+1) = w * V_m(t) + r1 * (X_m^* - X_m(t)) + r2 * (X^* - X_m(t)) \quad (4)$$

where $V_m(t)$ is the velocity at the preceding iteration, w is the inertia weight employed to adjust the influence of the previous particle velocities on the optimization process. X_m^* is the best historical position ever obtained by m, X^* is the best particle ever obtained during the algorithm, $r1$ and $r2$ are fixed parameters. We define the new position of the particle m as follows:

$$X_m(t+1) = X_m(t) + V_m(t+1) \quad (5)$$

In our work, the projection index represents the fitness function and the vector of projection defines a particle. In the first work, we used the classic version of the PSO with a modification of the notion of neighborhood in order to adapt the method to EPP. So, the practical application of the algorithm involves using X_l^* being the best particle in the neighborhood instead of X^*.

3.2 Tribes

Tribes is a hybrid PSO method based on a technique of stochastic optimization developed in [2] (see also [4]). This technique is a competitive algorithm which allows to find quickly local optima by investigating simultaneously several regions of the search space, generally local optima, before making a global decision.

In Tribes, particles are divided into several tribes, a metaphor for different sized groups of particles moving about in an unknown environment, looking for a "good" place. Each particle is evaluated by the fitness function (the projection pursuit index I). In each tribe, information links build a completely connected graph. Between tribes, links are looser, but the whole graph is still connected. This graph forms a structure able to diffuse and exploit information. This structure must be automatically generated and updated by means of creation, evolution and deletion of particles and tribes. Moving strategies of a particle, which indicate how a particle must modify its position, are based on "hyperspherical" probability distributions, which may be with or without noise, or independent Gaussian. The choice of these strategies is made depending on the short term history of the particle. This structuring will automatically induce the same purpose, namely explore several promising areas simultaneously, usually around local optima. In this part, we are going to define some notions allowing to understand the mechanism of this technique and give an algorithm describing its complete progress. Let us note that a particle is always defined as being a vector of projection.

The algorithm begins with one particle in a single tribe. Then it consists in creating and deleting particles and tribes. Along iterations, the position of the particles (value of the projection vector) is updated according to some strategies of displacement. Each time a tribe is created, links between particles are defined in order to make possible the transfer of information between tribes (in particular

x: *is the best position memorized by the particle during its course*
p: *is the best position memorized by the best particle of the generating tribe*
g: *is the best position memorized by the best particle of the swarm*
$nb\_iteration$: *the number of current iteration*
$Max\_iteration$: *The maximum number of iterations*
L: *the total number of information links*
$nb\_iteration = 0$; $L = 0$;

1. Create a first tribe formed of a single free particle
2. Estimate its fitness (the projection index)
3. Calculate $x = p = g$
4. $nb\_iteration++$
5. Create the second tribe, from the first tribe,
   consisted of a couple of free and stuffy particles
6. $L = 1$

for $nb\_iteration = 1$ to $Max\_iteration$ **do**
  Estimate fitness of every particle
  Calculate x, p, g for each particle
  Determine the quality of every particle and every tribe
  if Number of tribes < 3 and Both tribes do not improve their performance **then**
    Create the third tribe, from the first two tribes,
    formed of two pairs of free and stuffy particles
    $L++$
  else
    if $nb\_iteration = \frac{L}{2}$ **then**
      /* Do an adaptation */
      Create a new tribe
      Remove the worst monotribe of the swarm
      Remove the worst particle of each "good" tribe
      $L++$
    end if
  end if
end for

Algorithm 1. The Tribes algorithm

the best position of the particles in each tribe). Creating and deleting particles and tribes rely on measures of quality.

A particle is characterized by four possible qualities. It is labeled “good” if it has just improved its best performance (fitness value), “neutral” otherwise. A particle having the least good performance within its tribe is said “worse”, it is said “excellent” if its two last variations of performance (between successive iterations) are improvements.

The tribes themselves also receive the labels “good” or “bad”, depending on the number of good particles in the tribe. A tribe containing T particles is itself “good” only if $U() \leq G/T$, where G is the number of good particles in a tribe

and $U()$ is drawn from a standard uniform distribution. Otherwise the tribe is "bad". Good tribes, because they are doing well and presumably do not need as many particles, will remove one of their particles and only the worst of them, i.e. the particle with the highest value of I, if we assume that the projection index I is the function being minimized. When this occurs, any external links to the particle are re-assigned to the best performer in the tribe, i.e. the particle with the lowest value of I. In the case of a monoparticle tribe, the tribe itself is removed only if we are certain to keep in contact with all the tribes, i.e. all external links to the particle are reassigned to the external best particles. Bad tribes, on the other hand, presumably need more information, so each creates two new particles outside of its tribe and forms a link between the new particles and the best particle within the tribe. The set of all new particles created by all the bad tribes during one adaptation step forms a new tribe.

An adaptation is the realization of a deletion and/or a creation of particles, as described above. It occurs once at the beginning of the algorithm and then periodically as it progresses in order to propagate the information between particles. If, after adaptation, the number of links in the swarm is L, then adaptation will occur again after $L/2$ swarm iterations.

A particle adopts a strategy of movement according to its recent past and which looks like a local search. Three possibilities of variation of a particle's performance exist: deterioration (-), status quo (=) and improvement (+). The confinement of the particle in the search space is realized in the same way as in PSO but without velocity. Because the history of a particle includes two variations of performance, we find 9 possibilities of variation grouped in 3 strategies of movement according to the recommendations of Clerc [2]. The strategies are the pivot if the history of performance is $(--)$ or $(=-)$ or $(-=)$ or $(==)$, the disturbed pivot if the history is $(+=)$ or $(-+)$ or $(+-)$ and the local by independent gaussian if the history is $(=+)$ or $(++)$.

Pivot strategy: the new position of the particle is chosen at random according to an isotropic distribution centred on the pivot, for example a gaussian distribution.

$$x_d \leftarrow C_2 * \text{alea}(H_p) + C_3 * \text{alea}(H_g) \tag{6}$$

with p the best position memorized by the particle in the course of movement, g the best position stored by the best particle of the swarm, H_p, H_g two hyperspheres centred on p and g respectively and of the same radius equal to the distance $||g-p||$, $C_2 = \frac{I(p)}{I(p)+I(g)}$, $C_3 = \frac{I(g)}{I(p)+I(g)}$ and I the projection index.

Disturbed Pivot strategy: it is the same strategy as the previous one but with a noise. When we determine the new position, we modify it again according to a random gaussian noise. This noise will be very low if the performance of the particle is good. For every dimension of the space, we have:

$$\begin{cases} \sigma = \dfrac{I(p) - I(g)}{I(p) + I(g)} \\ b_d = \mathcal{N}_d((0, \sigma) \\ x_d = (1 + b_d)x_d \end{cases} \tag{7}$$

Local by independent gaussian strategies: the idea is to look for locally and only around the best position g known by the best particle. So, for every dimension d of the space, a coordinate close to the coordinate g_d of g is chosen at random according to a Normal law

$$x_d \leftarrow g_d + \mathcal{N}_d(0, |g_d - x_d|) \tag{8}$$

After the first iteration, if the situation does not improve, two particles will be generated, forming a second tribe. One of its two particles, called free, is generated anywhere in the search space according to a uniform distribution in the whole space and the other, said stuffy, is uniformly generated in a D-sphere of center g and of radius $||g - x||$, where g is the best position stored by the best particle of the swarm and x is the best position memorized by the best particle of the generating tribe. The idea of this generation is to intensify the search in a region which seems already interesting. At the next iteration, if neither of the two tribes improves its situation, each of the two tribes will generate another couple of new particles simultaneously, forming a new tribe of four particles, and the process will continue as described in the following algorithm. We note that as the number of links increases, the importance of the number of iterations between the two adaptations increases. Between two adaptations, the swarm then has more and more chances to find a solution.

The Tribes method is very useful in the resolution of the PP problem. This technique is efficient in most of the cases and allows the statistician to gain time by avoiding the tuning stage of the metaheuristic. Indeed, the statistician has to supply only the stopping time criterion and the objective function. Furthermore, [4] and [3] indicate that because the parameters are not assigned to their optimal values, the method converges early, resulting in being local optima on certain problems. Tribes is thus a very promising tool for the determination of several local optima which can reveal potentially interesting projections.

4 Application

Clustering is a set of statistical methods that separate data into classes (clusters) but, it is not clear how to assert that a data set contains well defined clusters. Our objective is to detect the presence of potential clusters in multidimensional data by using exploratory projections pursuit methods. We consider the two projection indices defined in section 2. In the present section, we give some results of the PSO and Tribes optimization methods applied to four data sets and demonstrate the interest to apply Tribes for the determination of the local optima of projection pursuit indices. The algorithms of these methods are implemented in language Java.

At first, we specify the number of particles for the PSO and the number of iterations for the PSO and Tribes. As it was recommended by Clerc [2], the PSO needs no more than 50 particles for small data sets. As regards to the number of iterations, we fixed it to 100 for the simulated and olive oil data for both methods. This value has been obtained by carrying out some preliminary runs

on each data set and checking the convergence of the indices to local optima. We ran 100 times each optimization algorithm on the different data sets and we present below some of the obtained results. In order to summarize the results in an efficient way, we draw the ranked values of the indices to each of the one hundred local optima with the projection vector corresponding to the best value of the index.

We present plots of the ranked values of the projection indices for the data sets using PSO and Tribes. We note that the number of launches can be increased during the exploration of big volumes of data. We also plot some histograms of the distributions of the projected data associated with local optima of the different indexes in order to visualize possible structure(s).

4.1 Simulated Data

We generated three data sets of $N = 1000$ observations and $P = 5$ variables. The observations are distributed according to various mixtures of standard normal distribution indicated as follows:

Normal2 contains two clusters of 500 observations with gaussian distribution $\mathcal{N}_5(\mu_i, I_5)$ with $i = 1, 2$ where $\mu_1 = (0, ..., 0)^T$ and $\mu_2 = (10, 0, ..., 0)^T$ are 5-dimensional vectors.

Normal4 contains four clusters of 250 observations with gaussian distribution $\mathcal{N}_5(\mu_i, I_5)$ with $i = 1, ..., 4$ where $\mu_1 = (0, ..., 0)^T$, $\mu_2 = (10, 0, ..., 0)^T$, $\mu_3 = (0, 10, 0, 0, 0)^T$, $\mu_4 = (0, 0, 10, 0, 0)^T$ are 5-dimensional vectors.

Normal10 contains ten clusters of 100 observations with gaussian distribution $\mathcal{N}_5(\mu_i, I_5)$ with $i = 1, ..., 10$ where $\mu_1 = (0, ..., 0)^T$, $\mu_2 = (10, 0, ..., 0)^T$, $\mu_3 = (0, 10, ..., 0)^T$, $\mu_4 = (0, 0, 10, 0, 0)^T$, $\mu_5 = (0, ..., 0, 10)^T$, $\mu_6 = -\mu_1$, $\mu_7 = -\mu_2$, $\mu_8 = -\mu_3$, $\mu_9 = -\mu_4$, $\mu_{10} = -\mu_5$ are 5-dimensional vectors and I_5 is the identity matrix.

The purpose of this example is to show the efficiency of the Tribes method in the detection of local optima which correspond to projections revealing the clusters structures of the data sets. On figure 1 we plot the 100 ranked values of the minimum kurtosis index for the simulated data with PSO (right curves) and Tribes (left curves). While the PSO method leads to small variability of the projection index values for the one hundred launches, Tribes supplies different local optima as soon as the interesting structure is complex and cannot be visualized on one dimension (Normal4 or Normal10). For the first two plots at the top, we tested the first data set Normal2 which contains two clusters. There is an unique interesting structure associated with an optimum index detected by the two methods. Both plots at the middle correspond to the data Normal4 which contain four clusters. We don't give the plots of the projections of the data but the structure in four clusters is detected by looking at the projections associated with the local optima corresponding to the three landings of Tribes. On the contrary, the PSO method does not allow to detect the four clusters.

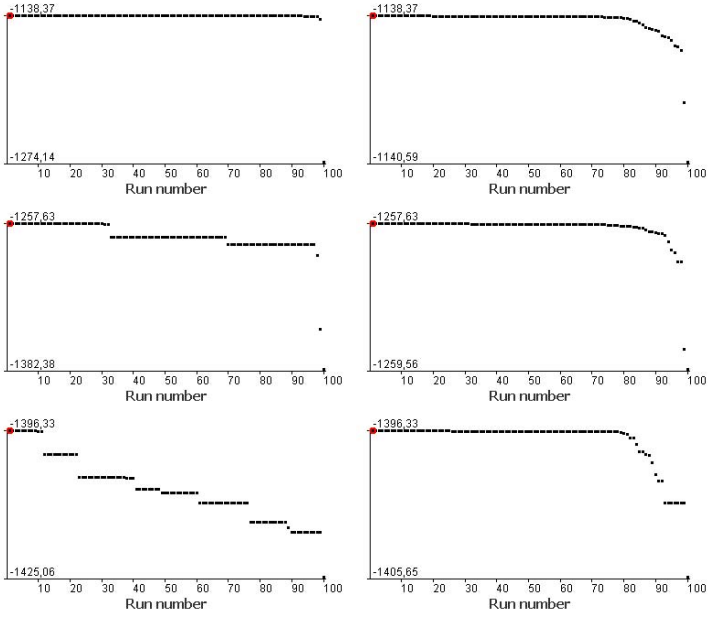


Fig. 1. Simulated data: Plots of the ranked values of the kurtosis index for the Normal2 (top curves), the Normal4 (middle curves) and the Normal10 (bottom curves) with PSO (right curves) and Tribes (left curves)

The index values associated to the last data set Normal10, which contains 10 clusters, are represented in the last two plots below. We notice that Tribes proposes several different local optima (see the landings) which represent various interesting projections. For these particular examples, the Friedman index gives the same results as the kurtosis index using both optimization methods.

4.2 Olive Data

The file consists in the percentage composition of $P = 8$ fatty acids found in the lipid fraction of $N = 572$ Italian olive oils. The 572 samples come from three different Italian regions subdivided themselves into nine areas. This data set has been analyzed by several authors in the context of exploratory multivariate analysis (see [6] and [1]). The structure of the data set is quite complex with nine clusters which have different shapes in an eight-dimensional space. Due to the large number of classes, discovering all of them by using one-dimensional EPP is challenging but the results we obtain clearly highlight a complex groups structure since several groups are detected by processing the data with the two proposed indices using Tribes.

As for the simulated examples, the plots (not given in the present paper) of the 100 ranked values of the minimum kurtosis index give different results

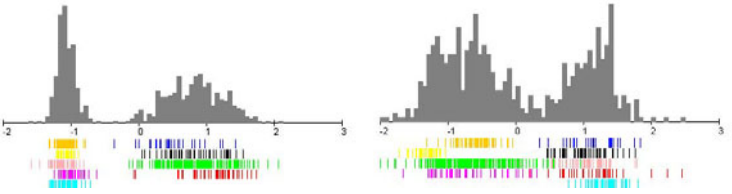


Fig. 2. Olive data: histogram corresponding to the global optimum (left figure) and a local optimum (right figure) for the minimum kurtosis index using Tribes

according to the optimization method. For PSO, it seems that there is only one potential interesting projection while we visualize at least two local minima for the kurtosis and the Friedman index with Tribes. In figure 2 we visualize two interesting projections corresponding to different local optima of the kurtosis index using Tribes method. On these two plots the data are split in two parts which correspond to different regions for the oils (see the clusters defined by the nine areas below the histograms). For the same method, the Friedman index gives different projections which separate other areas. As mentioned above, PSO yields a single interesting structure by optimizing any of both indices.

As illustrated in these small examples, EPP with the Tribes algorithm is a powerful tool for discovering clusters structures if present in the data. It would be interesting to test the proposed method on higher multidimensional datasets. Once EPP has revealed the presence of clusters, the data analyst may perform some clustering algorithm in order to define precisely the clusters.

5 Conclusion

In this paper, we used two metaheuristics (PSO and Tribes) to optimize two projection indices. We showed the performance of these methods using multidimensional data sets for the detection of groups. The important result of our study is the performance and the efficiency of Tribes method for projection pursuit. By using several simulations, we can easily obtain several local optima of the projection index susceptible to reveal interesting structures.

The difference between PSO and Tribes is that Tribes requires no parameter to settle. The statistician has only to define the objective function and the stopping criterion. A study of this method was led by [4] and [3] who found that Tribes converges very quickly to a local optimum which is not generally the global optimum. This characteristic, which the authors [4] and [3] consider as a drawback, motivates our choice and serves perfectly our objective.

Both Friedman and Kurtosis projection indices give good results on the considered examples. Concerning the computing time, we noticed that the kurtosis index is faster than the Friedman index. Although the evaluation number of the objective function is not the same for both methods (because the number of particles in the method Tribes is variable), we observed that Tribes is faster

than PSO. For the small-size data sets we consider, the time is unimportant for both methods and both indices but for very large data sets the kurtosis index together with the Tribes algorithm are recommended.

References

1. Caussinus, H., Ruiz-Gazen, A.: Exploratory projection pursuit. In: Govaert, G. (ed.) Data Analysis, Digital Signal and Image Processing, pp. 67–92. Wiley, Chichester (2009)
2. Clerc, M.: Particle swarm optimization. In: International Scientific and Technical Encyclopaedia. Wiley, Hoboken (2006)
3. Clerc, M., Cooren, Y., Siarry, P.: Optimisation par essaim particulaire améliorée par hybridation avec un algorithme à estimation de distribution. Journées doctorales MACS 5(1), 21–27 (2008)
4. Cooren, Y., Clerc, M., Siarry, P.: Performance evaluation of TRIBES, an adaptive particle swarm optimization algorithm. Swarm Intelligence 3, 149–178 (2009)
5. Cook, D., Swayne, D.F.: Interactive and Dynamic Graphics for Data Analysis. Springer, New York (2007)
6. Cook, D., Caragea, D., Honavar, H.: Visualization in classification problems. In: Antoch, J. (ed.) Computational Statistics (COMPSTAT), pp. 799–806. Springer, Berlin (2004)
7. Friedman, J.H.: Exploratory Projection Pursuit. J. Amer. Statist. Assoc. 82(1), 249–266 (1987)
8. Friedman, J.H., Tukey, J.W.: A Projection Pursuit Algorithm for Exploratory Data Analysis. IEEE Trans. Comput. C-23, 881–889 (1974)
9. Huber, P.J.: Projection pursuit. The Annals of Statistics 13(2), 435–475 (1985)
10. Jones, M.C., Sibson, R.: What is projection pursuit? (with discussion). J. Roy. Statist. Soc. A. 150, 1–36 (1987)
11. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Yuhui Shi. Morgan Kaufmann Publishers, San Francisco (1995)
12. Martinez, W., Martinez, A.: Computational statistics handbook with Matlab. CRC Press, Taylor and Francis Group (2001)
13. Nason, G.P.: Three-Dimensional Projection Pursuit. J. Roy. Statist. Soc. C 44, 411–430 (1995)
14. Peña, D., Prieto, F.: Cluster Identification using projections. J. Amer. Statist. Assoc. 96(456), 1433–1445 (2001)
15. Sun, J.: Some Practical aspects of exploratory Projection Pursuit. SIAM J. Sci. Comput. 14(1), 68–80 (1993)
16. Wu, J., Zhou, J., Gao, Y.: Support Vector Regression Based on Particle Swarm Optimization and Projection Pursuit Technology for Rainfall Forecasting. In: Int. Conf. on Computational Intelligence and Security, vol. 1, pp. 227–233 (2009)

Ant Colony Optimisation for Ligand Docking

Oliver Korb and Jason Cole

Cambridge Crystallographic Data Centre, Cambridge, UK
{korb,cole}@ccdc.cam.ac.uk

Abstract. In this work we propose a hybrid ant colony optimisation algorithm as an alternative search engine in the GOLD protein-ligand docking framework [4]. The approach treats the placement of a ligand molecule in the protein's binding site as a discrete assignment problem and a geometric point fitting procedure generates protein-ligand complex conformations from this representation. As in PLANTS [5,6], we combine this approach with a local search in the continuous search space of the objective function. Continuous solutions are finally reassigned to approximate solutions of the discrete assignment problem resulting in a high-performing optimisation approach. We discuss certain aspects of the hybridisation strategy including the integration of *heuristic information* into the search process and compare the performance to the *genetic algorithm* currently used in GOLD.

Keywords: ant colony optimisation, hybridisation, protein-ligand docking, heuristic information.

1 Introduction

Predicting protein-ligand complexes using computational methods is nowadays an integral part of structure-based drug design projects in pharmaceutical companies. Docking software such as GOLD, Glide or AutoDock are well-established approaches for predicting the conformation of small molecules in the binding site of a protein structure and for giving a crude estimate of the binding affinity [7]. Within the drug discovery process, these techniques are usually applied in either the screening stage, i.e. the identification of potential drug candidates, or the lead optimisation stage, where a specific molecular scaffold is structurally optimised. Almost all docking approaches treat the generation of protein-ligand complex conformations as an optimisation problem, where the variables to be optimised are given by the ligand's translational, rotational and torsional degrees of freedom and sometimes also by degrees of freedom in the protein. A so-called *scoring function* is used as the objective function in the optimisation problem and guides the sampling process for the generation of feasible complex conformations. The generated complex conformation can be compared to an experimentally determined structure in order to assess the accuracy of the prediction, if available. A common measure is the *root mean square deviation* (rmsd) of the ligand coordinates between the predicted and the experimentally determined structure.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 72–83, 2010.

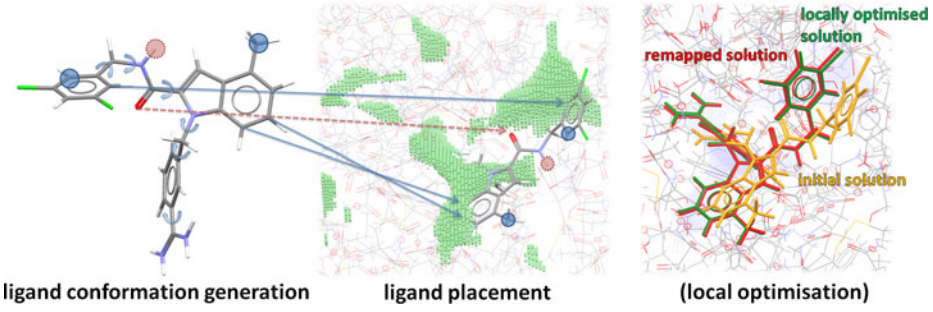


Fig. 1. Illustration of the fitting point based solution construction. A ligand conformation is constructed from the internal torsion angles (small arrows) and placed into the binding site using the assignment highlighted by solid (hydrophobic) and dashed (hydrogen bond) arrows. Circles mark unassigned atoms. The approach proposed in this work may additionally apply a continuous local optimisation step and remap the solution to the discrete assignment space.

Usually, an rmsd lower than 2 Å is considered to be a successful prediction. In a *virtual screening* experiment, hundreds up to millions of small molecules are docked and the optimised scoring function value is used to create a ranking of these compounds. In this context, the score is interpreted as the protein-ligand binding affinity and the top percentage of the ranked compounds is then tested experimentally for biological activity. Because large amounts of small molecules may need to be docked in a *virtual screening* experiment, the search time spent per ligand is critical. Besides the scoring function used, search time and therefore the optimisation performance have a direct impact on the final outcome of a *virtual screening* experiment. Thus, efficient and reliable optimisation approaches are needed.

1.1 Motivation

GOLD (Genetic Optimisation for Ligand Docking) is one of the most successful and widely used docking programs. An operator-based *genetic algorithm* (GA) is used for sampling favourable protein-ligand complex conformations guided by one of several scoring functions available. GOLD uses an island model and applies a niching technique to maintain a degree of diversity within the populations. The number of GA operations, as well as the mutation, crossover and migration probabilities are determined automatically by a heuristic that is based on the ligand structure to be docked and the polarity of the protein binding site. GOLD uses a unique fitting point based model to attempt to create solutions that contain chemically plausible protein-ligand interactions from the outset and during the docking process (see Figure 1).

Assignment of Fitting Points and Conformation Generation. Hydrogen bonding sites are used to assign fitting points from the ligand to the protein. For

each donor point in the ligand, an acceptor point is assigned at random. The acceptor point is either a suitable lone pair site on the protein or a 'dummy' site. Dummy sites indicate that a given donor should not be used in initially calculating the rotation and translation matrix for placing the ligand into the binding site. A similar approach to assignment is also used for acceptor points in the ligand to donor points in the protein, and for hydrophobic points in the ligand to hydrophobic points in the protein. Part of the GOLD chromosome encodes discrete torsion angles for all rotatable bonds in the ligand and the protein structure. Prior to placing the ligand into the binding site, the conformations of both the protein and the ligand are constructed from those values.

Ligand Placement and Scoring. Using the assigned pairs of points, 'Procrustes rotation' [1] is used to generate a transformation that maps the conformation into the binding site such that the fitting points overlap to the best degree. Following this initial step, the transformation is recalculated only using point pairs that lie within 2 Å of each other after the first rotation. This secondary step makes the algorithm focus on optimising the better interactions in the binding site. Finally, the generated protein-ligand complex is scored using one of several scoring functions.

After the GA has terminated, the best solution found is locally optimised using the Nelder Mead Simplex algorithm [8,10]. As stated above, GOLD can use different scoring functions for the optimisation process. The approach was initially designed for use with GoldScore, a force-field based scoring function. The fitting points were tailored to match favourable hydrogen bond geometries of this scoring function and, consequently, it is possible that the approach does not perform as well with other scoring functions. While a continuous local minimisation step is applied to the final GA solution, we were interested in designing a new algorithm applying a local minimisation step throughout the whole search process to overcome ligand placement restrictions imposed by the set of discrete fitting points. Our decision to use ant colony optimisation (ACO) [2] is based on the following reasoning. Firstly, as demonstrated for the ACO-based docking approach PLANTS [5,6], the combination of a discrete ACO algorithm with a continuous local search operator can be highly beneficial. Secondly, we were also interested in the integration of *heuristic information* into the search process. In this respect, ACO algorithms offer a unique way of directly biasing the solution construction step towards solutions exhibiting certain characteristics.

2 Materials and Methods

2.1 Problem Representation

For the proposed approach, we use two different problem representations in tandem. The *discrete* representation used for the ACO algorithm determines the ligand translation and orientation by defining an assignment of ligand fitting points to protein fitting points as described before, while the internal ligand and protein conformation are encoded by selecting the respective torsion angles. The *continuous* representation uses the 6 rigid-body degrees of freedom for the definition of

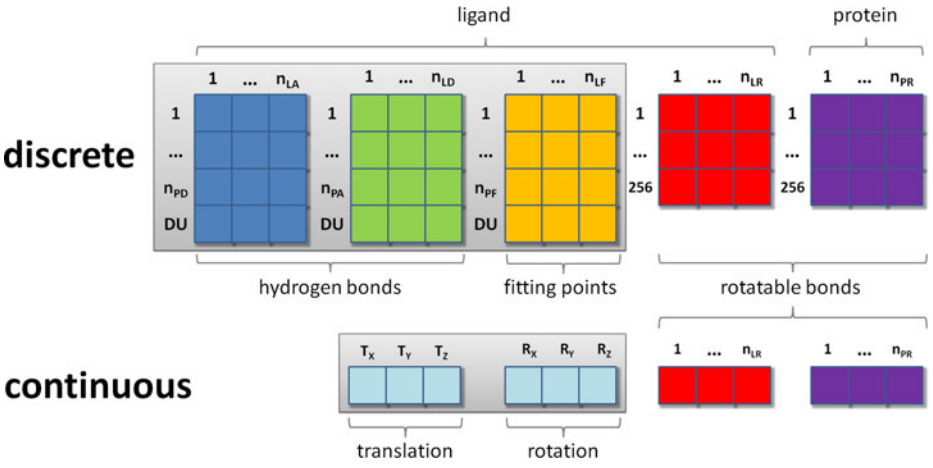


Fig. 2. Problem representations. The upper part shows the discrete representation as an assignment problem, while the lower part illustrates the continuous representation used in the local optimisation algorithm. In both cases, the ligand's orientation is determined by the degrees of freedom highlighted by a grey background.

the ligand translation and orientation. This representation is suitable to be used by a continuous local minimisation procedure.

Discrete. An illustration of the discrete problem representation can be found in the upper part of Figure 2. The problem dimension is given by $n = n_{LD} + n_{LA} + n_{LF} + n_{LR} + n_{PR}$, where n_{LD} is the number of ligand donors, n_{LA} the number of ligand acceptors, n_{LF} the number of ligand hydrophobic fittings points, n_{LR} the number of ligand rotatable bonds and n_{PR} the number of protein rotatable bonds. The ligand's orientation is defined by the assignment of ligand donors, acceptors, and hydrophobic atoms to the protein fitting points. Each of the assignment vectors $\boldsymbol{a}_i$ has n_i entries, where $i \in \{1, \cdots, n\}$, and a specific entry $a_{ij} \in \{0, 1\}$ is one if variable i is assigned to entry j or zero otherwise. For a ligand donor vector $\boldsymbol{LD}_i$, the number of vector entries corresponds to $n_i = n_{PA} + 1$, where n_{PA} is the number of protein acceptors. The vector is extended by one additional entry to allow the donor to be left unassigned (marked as 'DU' in Figure 2). Similarly, for a ligand acceptor vector $\boldsymbol{LA}_i$ and a ligand fitting point vector $\boldsymbol{LF}_i$ the number of vector entries correspond to $n_i = n_{PD} + 1$ and $n_i = n_{PF} + 1$, respectively, where n_{PD} and n_{PF} are the number of protein donors and protein hydrophobic fitting points, respectively. For a ligand or protein torsion angle vector, i.e. $\boldsymbol{LR}_i$ and $\boldsymbol{PR}_i$, the number of vector entries corresponds to $n_i = 256$, following the 8 bit encoding used in the chromosome of the GA. In a valid assignment for each column vector $\boldsymbol{a}_i$, exactly one entry j has to be one, i.e. $\forall_{i=1}^{n} \sum_{j=1}^{n_i} a_{ij} = 1$. In the ACO approach presented in this work, each assignment entry a_{ij} has an associated pheromone intensity $\tau_{ij} \in \mathbb{R}$, i.e. a desirability of choosing this value in the solution construction process. For

entries of a ligand donor or acceptor vector, additionally an *a priori* desirability $\eta_{ij} \in \mathbb{R}$ may be assigned as will be explained later.

Continuous. The continuous representation is illustrated in the lower part of Figure 2. In contrast to the discrete encoding, the ligand orientation is defined by 6 floating point values, i.e. 3 for the ligand's translation and 3 for the rotation. Torsion angles in the protein and the ligand are also represented as single floating point values. Hence, in the continuous case the problem dimension is $n = 6 + n_{LR} + n_{PR}$.

The scoring functions in GOLD return positive values for favourable solutions and, hence, the presented approach models the problem as a maximisation problem. Throughout the rest of this paper, we will denote the assignment of a solution s by s^a, the protein-ligand complex conformation by s^c and the objective function value by s^f.

2.2 Algorithm

In general, the ACO algorithm (see Algorithm 1) follows the latest PLANTS version [6], but using the problem representation of GOLD described above.

Initialisation and Number of Iterations. Before each docking run, the fitness function, the *heuristic information* and pheromone distribution are initialised. For each ligand the algorithm is executed for a certain number of iterations,

$$iterations = autoscale \cdot (100 + 25 \cdot lrb + 5 \cdot lha), \tag{1}$$

algorithm 1. GOLD$^{\text{ACO}}$

```
InitialiseParametersAndPheromones()
for i = 1 to iterations do
  for j = 1 to m do
    s_j^a ← CreatePheromoneBasedAssignment()
    s_j^{a, c} ← PerformProcrustesRotation(s_j^a)
    s_j^f ← EvaluateObjectiveFunction(s_j^c)
  end for
  s_ib ← GetIterationBestSolution()
  if useLocalSearch then
    s_ib^{c,f} ← LocalSearch(s_ib^c)
    if remapContinuousSolution then
      s_ib^a ← RemapSolution(s_ib^c)
    end if
  end if
  UpdatePheromones(s_ib^a, s_ib^f)
  if diversificationCriteriaMet then
    ApplySearchDiversification()
  end if
end for
return best solution found
```

where *autoscale* is a scaling factor and *lrb* and *lha* correspond to the number of ligand rotatable bonds and heavy atoms, respectively. Consequently, more search time is spent on large and flexible ligands than on small and rigid ones.

Solution Construction. For each of the m artificial ants in the colony, an assignment s^a is constructed taking *heuristic information* and the already deposited pheromone into account. The probability for value j being assigned to variable i is given by

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l=1}^{n_i} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, \tag{2}$$

where τ_{ij} is the pheromone intensity for this value and η_{ij} is the *heuristic information*. Solution construction is biased towards ligand assignments that exhibit strong polar interactions (see Figure 4). The value of η for a specific donor / acceptor pair is given by the GoldScore hydrogen bond energy. We set $\eta = 1.0$ for the choice of not assigning a ligand donor or acceptor. The influence of the pheromone and *heuristic information* is scaled by parameters α and β, respectively. As will be shown later, we examined the effect of varying β, but set $\alpha = 1$ for all experiments. Each fitting point is used (at most) once in each solution, thus avoiding duplicate assignments. When a complete assignment has been constructed, the geometric fitting procedure calculates a ligand placement s^c and the objective function value is evaluated. Note that the solution construction process is repeated until a valid structure is found.

Local Search and Remapping. If local search is used, the Nelder Mead simplex algorithm [8] locally optimises the ligand conformation of the iteration-best solution, s_{ib}^c, resulting in a new conformation s_{ib}^c and a new objective function value s_{ib}^f. We follow the implementation described in [10]. The initial continuous representation is created by converting the discrete protein and ligand angles to a floating point representation. The values for translational and rotational degrees of freedom are set relative to the transformation matrix that resulted from the geometric fitting procedure. The offset values for the construction of the initial simplex are set to 2 Å for translational degrees of freedom and 60° for rotational and torsional degrees of freedom. The first solution of the initial simplex is given by the original one, all other ones by adding the specific offset values to the original solution. For example, the second solution is obtained by adding 2 Å to the T_X component of the original solution. The simplex algorithm is terminated if either a maximum of 1500 simplex iterations is reached or the fractional range between the worst and best solution solution of the simplex is lower than 0.001 [10]. If the option for remapping the continuous solution to the discrete assignment space is activated, the following procedure is executed, which updates the iteration-best assignment s_{ib}^a. Protein and ligand torsion angles are discretised to their nearest 8 bit representation. For ligand acceptor and donor atoms the nearest protein donor and protein acceptor fitting points are located, respectively. If, for a specific atom, no fitting point is located within 1.5 Å, the atom remains unassigned. A similar procedure is used for the ligand's

hydrophobic atoms. The nearest protein fitting point within 0.2 Å is assigned and the ligand atom remains unassigned if no such point is found.

Pheromone Update. First, the lower and upper pheromone limits τ_{min} and τ_{max} are recalculated. While a single value for τ_{max} is used as proposed in the original publication of $\mathcal{MM}$AS [11], τ_{min_i} needs to be calculated per variable i as the number of values n_i for each pheromone vector differs. We set p_{best}, the probability of reconstructing the best solution assuming that the colony has converged, to a value of 0.5, refer to PLANTS [5] for details. In each iteration, already deposited pheromone evaporates and the iteration-best solution s_{ib} updates the pheromone distribution according to

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + I_{ij}(s_{ib}, t)\Delta\tau(s_{ib}, t), \tag{3}$$

where

$$\Delta\tau(s,t) = \begin{cases} s^f + 20.0 \text{ if } s^f > -20.0 \\ 0 \qquad \text{otherwise} \end{cases} \tag{4}$$

In this equation, ρ is the evaporation rate and I_{ij} is an indicator function returning 1 if value j is used for variable i in an assignment s^a. Note that the iteration-best solution must have a fitness better than -20 units to be allowed to update the pheromone distribution. Worse solutions make no contribution. Additionally, if more than 5 iteration-best solutions are worse than the best solution since the last search diversification, s_{db}, solution s_{db} also updates the pheromone distribution in proportion to its fitness s^f_{db}. Finally, all pheromone values are adjusted according to the lower and upper pheromone limits.

Search Diversification. We use a similar search diversification schedule as described for PLANTS. If the fitness values of 10 iteration-best solutions differ by less than $0.02 \cdot |s^f_{gb}|$, where s_{gb} is the global best solution found, a search diversification is carried out. This can either be a proportional pheromone smoothing with a smoothing factor of 0.5 or a restart of the algorithm. In general, the diversification schedule consists of 3 pheromone smoothings followed by a restart.

Finally, when the algorithm terminates after a fixed number of iterations, the best solution found is returned. An illustration of the steps carried out in one iteration of the approach can be found in Figure 1.

2.3 Parameter Optimisation and Validation

The approach has been trained on 10 protein-ligand complexes from the CCDC / Astex data set [9] (Protein Data Bank codes 1a4q, 1atl, 1c5c, 1mmq, 1mrg, 1ppc, 1tnl, 1xie, 2cpp and 4dfr) covering a wide range from small and rigid to large and flexible ligands. We tested the values 10, 25, 50 and 100 for the number of ants, 0.05, 0.1, 0.2 and 0.3 for evaporation rate ρ, 0.25, 0.5 and 1.0 for the scaling parameter *autoscale*, 0, 1, 2 and 5 for parameter β and investigated the effect of turning the simplex optimisation and the solution remapping on and off. Solution remapping was only considered if the simplex optimisation was turned on. In total, this resulted in 576 unique parameter settings, which

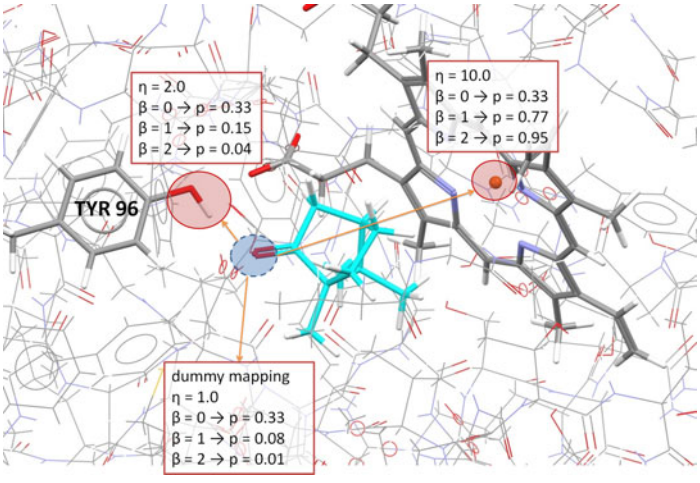


Fig. 4. Problematic example for the incorporation of *heuristic information* (Protein Data Bank code 2cpp). For further explanations see the text.

were used to dock the training set. Each experiment was repeated 50 times and average docking times and fitness values were recorded. For each complex, the spherical binding site definition as given in the CCDC/Astex data set [9] was used. The final parameter settings were tested on the independent Astex diverse test set [3] containing 85 high-quality protein-ligand complexes. For each of these complexes, the binding site was defined by all protein atoms within 6 Å from any ligand heavy atom as given in the crystal structure conformation. For the GA, standard settings were used, i.e. the parameter values for cross-over, mutation and migration operators were set automatically in dependence of the protein-ligand complex and the value of *autoscale* specified. Per experiment, the GA was allowed to perform at most 10 runs and *early termination* was switched on. Hence, if the rmsd of the 3 top-ranked solutions is within 1.5 Å to each other, the search is terminated. We used scoring function CHEMPLP [6] and all experiments were carried out on an Intel Xeon CPU E5420, 2.50 GHz.

3 Results and Discussion

Parameter Optimisation. The results obtained for the parameter optimisation process are presented in Figure 3. In all plots, the x-axis reports the average search time (in seconds), while the y-axis reports the average fitness or success rate over 50 independent experiments.

We will first discuss the effect of parameter *autoscale* as well as the use of local search and the solution remapping step (see Figure 3a). The dashed polygon separates parameter configurations not using local search (inside) from those using the simplex optimisation step. Likewise, the dashed line, separates parameter

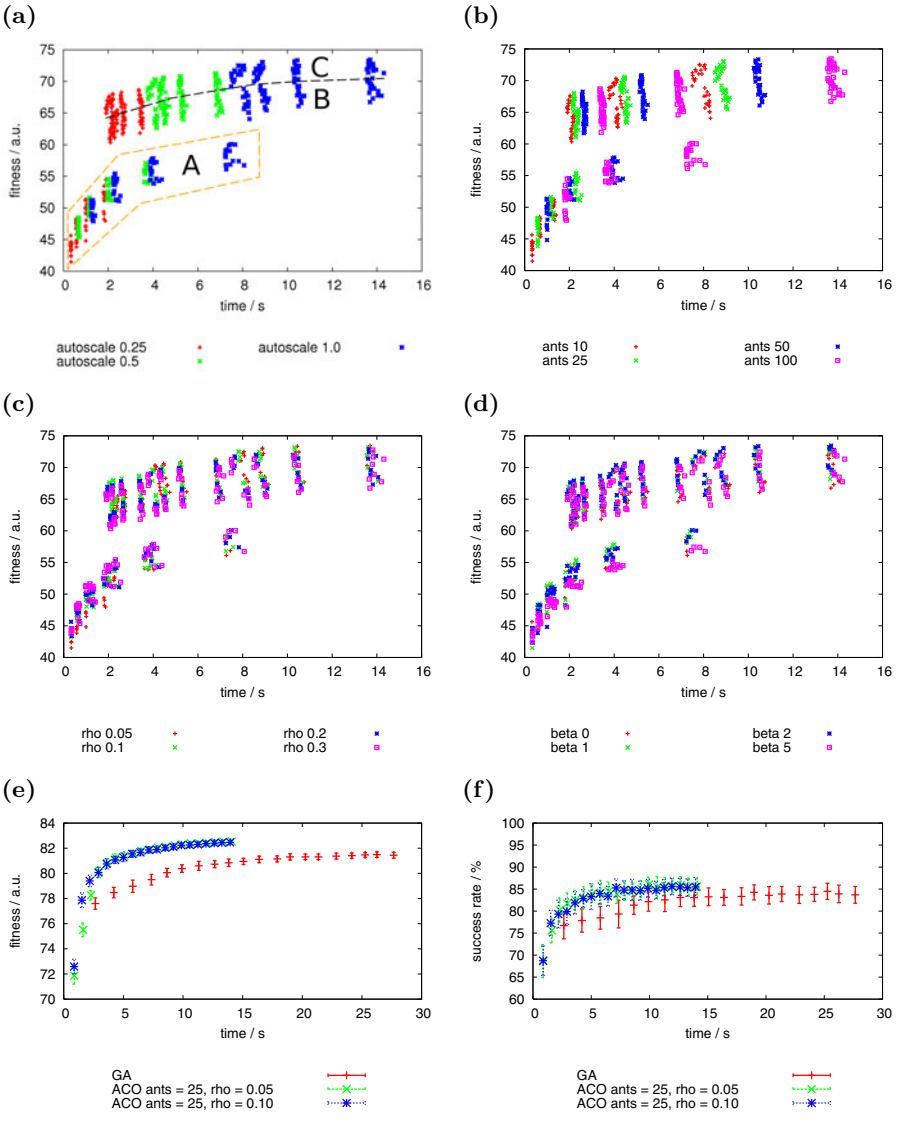


Fig. 3. Results obtained for the parameter optimisation (a-d) and the docking experiments (e-f). In each plot the x-axis reports the average docking time per complex, while the average fitness values reached is shown on the y-axis, except for (f) in which case the average success rate is presented (all results averaged over 50 independent runs). In (a) the dashed line separates configurations using the remapping technique (above) from configurations not using it (below). All parameter configurations contained in the dashed polygon use no simplex optimisation. In (e) and (f) additionally the standard deviations are plotted. For further explanations see the text.

configurations using the solution remapping technique (above) from those that don't (below). This results in the three different algorithmic variants (A) local search deactivated, (B) local search activated / remapping deactivated and (C) local search activated / remapping activated. As expected, variant (A) using the ACO approach performs worse than variants (B) and (C) using the additional local search step. Interestingly, the worst parameter configuration for variant (B) reaches approximately the same average solution quality as the best-performing setting for (A) in less than a third of the search time. Algorithm variant (B) essentially learns favourable starting positions for the local search as the assignment of the iteration-best solution, s^a_{ib}, before the execution of the local search is used to update the pheromone distribution. However the amount of pheromone updated is proportional to the solution quality obtained after application of the local search, s^f_{ib}. Finally, activating the remapping of the locally optimised solution to a discrete assignment and using this to update the pheromone distribution, i.e. variant (C), again improves the average solution quality compared to variant (B). With respect to parameter *autoscale* scaling the number of iterations, a clear separation of clusters for the three settings studied can be observed for algorithm variants (B) and (C), while for (A) there is a strong dependency on the number of ants used. As expected, a higher solution quality is obtained for higher settings of *autoscale*, however at the cost of an increased search time. According to Figure 3b, the sub-clusters for each setting of *autoscale* correspond to the different settings for the number of ants in the cases of (B) and (C). Again, search time and solution quality increase with the colony size. In the case of variant (A), a lower setting of *autoscale* combined with a larger number of ants may find higher quality solutions in a shorter time. The influence of evaporation factor ρ is shown in Figure 3c. Approach (A) generally favours higher evaporation rates, i.e. values of $\rho = 0.3$ and $\rho = 0.2$ seem to perform best. In contrast, when using local search like in the variants (B) and (C) exploration seems to become more important and values of $\rho = 0.05$ and $\rho = 0.1$ are among the best-performing settings. Finally, we also investigated the effect of different settings for parameter β scaling the influence of the *heuristic information*. As can be observed in Figure 3d, for approach (A) lower settings of β seem to be more favoured, especially for very quick search settings. This behaviour can be explained by the composition of the training set. Protein-ligand complex 2cpp was included for the purpose of limiting the influence of the *heuristic information*. The crystal structure of this complex is visualised in Figure 4. In the solution construction process, the ligand acceptor atom (marked with a dashed circle) can in principle be assigned to the receptor's hydroxyl donor of the tyrosine (residue TYR96), the iron atom in the porphyrin system (right side) or it can be left unassigned. The probabilities of each being selected in the first iteration of the solution construction process are shown as a function of β. The *heuristic information* for the three choices are $\eta = 2$ for selecting the hydroxyl donor, $\eta = 10$ for selecting the iron atom and $\eta = 1$ for not assigning the acceptor at all. Note that in the crystal structure the ligand acceptor forms a hydrogen bond to the hydroxyl of the tyrosine and not the iron atom which contradicts the *heuristic*

information. However, the objective function is able to predict this complex correctly, i.e. the global maximum is similar to the crystal structure. If β is zero, the selection probabilities for all three choices are $\frac{1}{3}$, while the construction process is highly biased towards the construction of solutions using the iron atom (the selection probability increases from $\frac{1}{3}$ to 0.77 and 0.95, for $\beta = 0$, 1 and 2, respectively). Thus, for high values of β nearly all of the constructed solutions will use the iron atom assignment and quick search settings will possibly not sample the correct ligand pose. For longer search settings, higher values for β become more beneficial again as the probability of generating an assignment which may be suboptimal according to the *heuristic information* increases. In general, algorithm variants (B) and (C) seem to favour settings of $\beta = 2$ independent of the search time. The standard settings we derive from these observations are to use algorithm variant (C), $m = 25$ ants, an evaporation rate of $\rho = 0.05$ or $\rho = 0.1$ and a setting of $\beta = 2$. These settings were finally tested in independent docking experiments and compared to the GA available in GOLD.

Docking. Both the new ACO-based approach presented in this work and GOLD's GA have been used to dock the 85 protein-ligand complexes of the *Astex diverse* set [3]. We ran both methods for 20 different settings of parameter *autoscale* starting from *autoscale* $= 0.1$ using a step-size of $\Delta = 0.1$. Figures 3e and 3f plot the average fitness and success rate on the y-axis against the average search time needed per ligand on the x-axis averaged over 50 independent runs. As can be observed in Figure 3e, the new approach exhibits a superior optimisation performance, reaching on average much higher fitness values at shorter search times compared to the GA. Even for very long search times, the GA is not capable of reaching the same average solution quality. As expected, a higher evaporation rate of $\rho = 0.1$ is favourable for faster search settings, while the differences in performance are negligible for longer search settings. The average pose prediction success rates, i.e. reproducing the top-ranked solution within an rmsd of 2 Å compared to the experimentally observed ligand structure, are presented in Figure 3f. The ACO-based approach still outperforms the GA with average success rates of up to 85 %, but the differences are less pronounced. This can be explained by the fact that although the GA may not necessarily find optimal solutions with respect to the fitness value, it still produces geometrically and chemically feasible ligand placements.

4 Conclusions and Future Work

In this work, we have presented an efficient hybrid ant colony optimisation approach to the protein-ligand docking problem. The new approach benefits from a combination of the discrete fitting point based representation as used by the GA in GOLD with a continuous local optimisation procedure. Additionally, we demonstrated the successful integration of *heuristic information* into the solution construction process. Docking experiments confirmed the high performance of the new approach, which was able to find, on average, higher fitness values at much shorter search times compared to the GA as currently implemented in

GOLD. While we only biased the hydrogen bond assignment in the pheromone-based solution construction process in this work, we will also try to incorporate *heuristic information* for non-hydrogen bonding atoms in future research efforts. We plan to use data from either precomputed scoring function grids or interaction databases like *SuperStar* [12] to bias the placement of ligand atoms to appropriate regions of the receptor. Ultimately, we will also investigate possibilities of integrating these algorithmic improvements into the GA.

Acknowledgments. The authors thank Drs Simon Bowden and Colin Groom for a careful reading of the manuscript. O.K. was funded through a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD).

References

1. Digby, G.N., Kempton, R.A.: Multivariate Analysis of Ecological Communities, pp. 112–115. Chapman and Hall, London (1987)
2. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
3. Hartshorn, M.J., Verdonk, M.L., Chessari, G., Brewerton, S.C., Mooij, W.T.M., Mortenson, P.N., Murray, C.W.: Diverse, high-quality test set for the validation of protein-ligand docking performance. J. Med. Chem. 50(4), 726–741 (2007)
4. Jones, G., Willett, P., Glen, R.C.: Molecular Recognition of Receptor Sites Using a Genetic Algorithm with a Description of Desolvation. J. Mol. Biol. 245, 43–53 (1995)
5. Korb, O., Stützle, T., Exner, T.E.: An ant colony optimization approach to flexible protein-ligand docking. Swarm Intelligence 1(2), 115–134 (2007)
6. Korb, O., Stützle, T., Exner, T.E.: Empirical Scoring Functions for Advanced Protein-Ligand Docking with PLANTS. J. Chem. Inf. Model. 49, 84–96 (2009)
7. Moitessier, N., Englebienne, P., Lee, D., Lawandi, J., Corbeil, C.R.: Towards the development of universal, fast and highly accurate docking/scoring methods: a long way to go. British Journal of Pharmacology 153(S1), S7–S26 (2008)
8. Nelder, J.A., Mead, R.: A simplex method for function minimization. Computer Journal 7, 308–313 (1965)
9. Nissink, J.W.M., Murray, C., Hartshorn, M., Verdonk, M.L., Cole, J.C., Taylor, R.: A new test set for validating predictions of protein-ligand interaction. PROTEINS: Structure, Function, and Genetics 49(4), 457–471 (2002)
10. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press, Cambridge (1992)
11. Stützle, T., Hoos, H.H.: $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. Future Generation Computer Systems 16(8), 889–914 (2000)
12. Verdonk, M.L., Cole, J.C., Watson, P., Gillet, V., Willett, P.: Superstar: improved knowledge-based interaction fields for protein binding sites. J. Mol. Biol. 307(3), 841–859 (2001)

Antbots: A Feasible Visual Emulation of Pheromone Trails for Swarm Robots

Ralf Mayet, Jonathan Roberz, Thomas Schmickl, and Karl Crailsheim

Karl-Franzens University Graz, Artificial Life Lab of the Department of Zoology, Graz, Austria
ralf.mayet@uni-graz.at, jonathan.roberz@rwth-aachen.de, thomas.schmickl@uni-graz.at

Abstract. In this paper we present an experimental setup to model the pheromone trail based foraging behaviour of ants using a special phosphorescent glowing paint. We have built two custom addons for the e-puck robot that allow for trail laying and following on the glowing floor, as well as a way for the robots to mimic the ants capability of using polarization patterns as a means of navigation. Using simulations we show that our approach allows for efficient pathfinding between nest and potential food sources. Experimental results show that our trail and sun compass add-on boards are accurate enough to allow a single robot to lay and follow a trail repeatedly.

Keywords: swarm robotics, ant foraging, pheromone trails, biomimicry.

1 Introduction

Ants have the ability to find the shortest possible path between food sources and their nest collectively by laying pheromone trails on the ground [1,14]. These pheromones have shown to be involved specifically in the recruitment and navigation of ants between food sources and the nest. The foraging behaviour has long been of particular interest not only in the field of biology, but also in swarm robotics [4]. These kinds of collective abilities can be applied to real-life problems such as traffic, route-planing or the travelling salesman problem as well [5].

The collective behaviour of social insects and stigmergy-based communication remain to have a strong influence on the field of collective robotics. There have been several approaches to model the pheromone-based trail laying and trail following behaviour of ants in experimental settings using robots.

By means of chemical sensors and alcohol-depositing robots [10]. This is a very realistic imitation of the pheromone-based trails of ants. However, the chemical sensors used in this setup and the combination of robotics and substances such as alcohol have been shown to be very unreliable and not very practical.

Drawing lines onto the floor using pen and paper [12]. In this scenario each robot is equipped with a pen, with which it is able to draw solid thin lines onto the ground. Although a decay of these trails is archieved by using a special kind of disappearing ink, the trails layed by these robots remain thin in comparison

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 84–94, 2010.

to the robots. This does not provide a close analogy to the biologically inspired behaviour of ants.

Laying trails of heat [9]. This method promises an extremely flexible way to model the foraging behaviour of ants by laying trails of residual heat onto normal surfaces such as carpets or tiles. One problem is that the electrical generation of heat is not possible even on bigger mobile robots because of constraints in battery power. The researchers stored heat in the form of paraffin wax to lay trails instead. This presents an additional difficulty for experimental use and dynamically adjusting the strength of the trail is not possible.

Using robot-tracking and a projector setup, in which each robot is able to lay trails by being tracked using a camera suspended above the arena [7]. A computer superimposes 'virtual pheromones' by projecting them onto the arena floor. This system does not present a fully autonomous way for the robots to lay and follow trails, and a a central unit, an external computer, is needed. However, this system provides a very flexible way to modify parameters of the pheromones, such as decay and diffusion.

Emitting ultraviolet light onto a phosphorescent paint, and thus laying luminous green trails on the arena floor. This method of modelling ant trails has been published for use in an artistical context [2]. In this setup, the arena floor is coated with a special phosphorescent glow-paint that glows in the dark for several minutes after being stimulated by an external UV light source. By attaching UV-LEDs to the mobile robots, they can leave glowing trails on the ground. The idea is that because of the constant decay in brightness, the green glow that eminates from the floor can be seen as an analogy to the evaporating pheromones ants utilize in their trail following.

In this paper we extend and improve on the idea of using glow-paint to mimic ant trails. It presents a completely autonomous way for the robots to lay trails. Using specially developed sensors and actuators allows us to combine the glowing floor and robots in a unique and reliable way.

2 Materials and Methods

2.1 Base Robot: E-Puck

The e-puck robot [3] (see Fig. 2), was developed at the EPFL in Switzerland. It utilizes a Microchip dsPIC microcontroller at its core, and has two stepper motors for actuation. Several red and green LEDs placed in and around the casing of the robot can be used to quickly display what state the robot is in at a particular time. The e-puck features a number of different sensors, most important for our experiments are eight proximity sensors and a color CMOS camera. The proximity sensors are infrared receivers and transmitters placed around the body. They allow the measurement of distance between the robot and obstacles. Additionally they can differentiate between obstacles like walls and other robots by means of active and passive sensing. The camera on the e-puck has a theoretical resolution of 640×480 pixels, but only a part of this

frame can be grabbed at runtime with a high enough framerate (40×40 pixel color image at approx. 4 frames per second).

2.2 Glow-Paint Floor

The synthetic raisin paint used to coat the arena floor contains small grains of phosphorescent material that react instantly to ultraviolet light and glow in the dark with a characteristic decay time T, as the intensity decreases like $I(t) = I_0 \exp(-t/T)$. In our experiments we have used the onboard camera tilted downwards for trail detection and trail following. Figure 1 presents intensity measurements over time for the onboard e-puck camera from the point of view of the robot. Figure 2 shows the robots' field of vision on the floor. The line-following algorithm is based on a line-following Braitenberg vehicle.

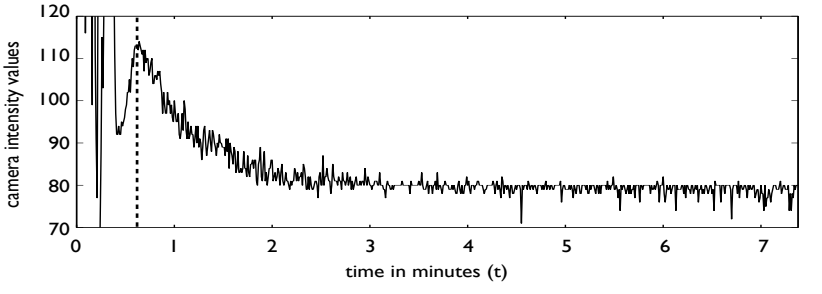


Fig. 1. Intensity measurements from the onboard e-puck camera over the course of approx. 7 minutes for the green color channel, directly polled from the robot, while standing still and the camera is pointed at one spot on the coated floor. Ambient light was at complete darkness. Before the dashed line ($t = 0$ to $t \approx .5$) the floor is being exposed to direct UV-LED light, explaining the irregular peaks, and is at its maximum glowing capacity shortly thereafter at $t \approx .6$. This curve combines the characteristic decay of the phosphorescent paint and the camera's response curve.

2.3 Nest, Food and an Artificial Sun as a Navigational Aid

Analogous to nest and food sources in ant foraging, we have chosen LED light sources with different colors for our experiments. The robots can distinguish between the objects by comparing the different color channels of the camera. The nest has been chosen to be red, and food sources are blue. Each one is made of transparent plastic covers with LEDs underneath.

Ants derive directional information from polarization patterns to navigate back to their nest from food sources [8]. This behaviour has been described for the desert ant *Cataglyphis fortis* in great detail [6]. To emulate this ability we have chosen a red light source positioned in a corner above the arena. Figure 3 shows a diagram of our experimental setup.

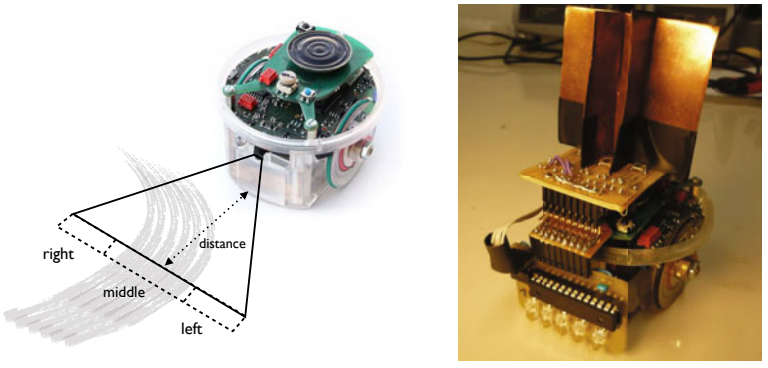


Fig. 2. Left: Diagram showing how far and how wide the field of view of the camera is on the floor. Dashed lines represent the sectors used for the line following algorithm: The robot calculates the brightness of the green color channel for each of the three sectors. When the leftmost sector is the brightest it turns right and vice versa. If the middle sector is the brightest it moves straight ahead. Distance from robot is approximately 4.5 centimeters. **Right**: e-puck robot equipped with our trail-laying (front) and sun compass extension board (top).

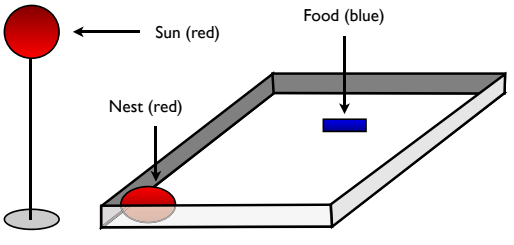


Fig. 3. This diagram shows the experimental setup used in our experiments. Depicted are the artificial sun suspended above the arena, the nest and a foodsource.

Each robot can detect from which direction this emulated sunlight is coming from using a specially designed 'sun compass' add-on board (see Fig. 2 right). In our experiments the nest has been placed in the same corner as this artificial sun, so the robots simply have to drive towards it to find their nest.

2.4 Add-on Boards: Trail-Laying and Sun Compass Extension

To utilize the experimental setup described above, the e-puck base robot needed several extensions (see Fig. 2). A special circuit board was added to provide the robot with five UV-LEDs pointed directly to the floor. The add-on uses a dedicated ATMega8 MCU [13] for independent control over brightness for each LED using pulse-width modulation. This board is attached on the back of the

robot for a minimum distance of LEDs to the floor and thus the maximum amount of energy reaching the floor.

An additional board featuring six photodiodes that measure light intensity (red wavelengths in particular) is attached to the analog-digital comparer of the MCU. It is located on top of the robot and the sensors point upwards. Three small rectangular sheets of copper have been added to allow the 'artificial sun' to cast shadows on the sensors that don't point towards the light (see Fig. 2 right). By calculating which sensor has the highest value the robot is then able to face the artificial sun and readjust its direction during its trip to the nest, if necessary.

2.5 Simulation Using Netlogo

In order to explore parameters and the efficiency of our experiment, we have built a simulator (see Fig. 4) using the multi-agent programmable modeling environment *NetLogo* [11], developed at the Northwestern University in Illinois. In NetLogo *agents* are the equivalent of robots, and the arena is made up of tiles, so called *patches*.

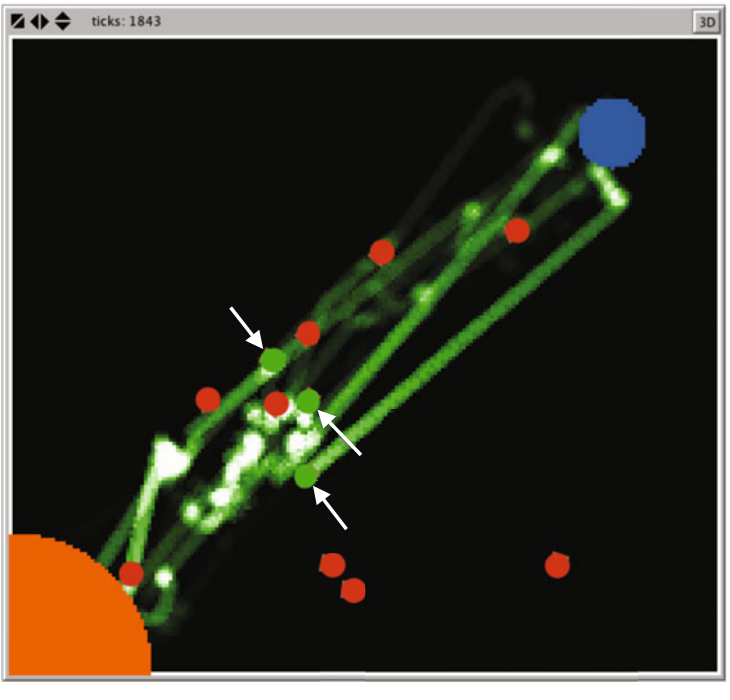


Fig. 4. Screenshot of a simulation run. The nest (orange) is located on the bottom left, a foodsource (blue) in the top right. Agents are either red (searching for food) or green (searching for the nest, marked with arrows here) depending on their state. Green trails show the pheromone paths between food and nest.

To resemble our experimental setup as closely as possible, we have programmed the eight proximity sensors of the e-puck robot into the simulation. These sensors have been implemented actively and passively in order to be able to differentiate between walls and other agents and to allow for a sufficient approximation of the obstacle avoidance of real robots. Our sun compass sensor was also implemented in the simulator by calculating where the nest is in relation to the agent. The resulting angle is categorized into six different general directions to approximate the sun compass. The agents leave a predefined amount of virtual pheromones on the patch they are currently on. The simulator then computes different shades of green in order to visualize the amount of pheromone on each patch for every time step. Nest (bottom-left) and foodsource (top-right) are orange and blue circles respectively. The camera, used for trail-following, is emulated by reading out pheromone levels (shades of green) in front of the individual agent at three different angles.

In contrast to most other ant trail multi-agent simulations, ours models the actual sensoral attributes of the e-puck robots, and actual properties of the glowing floor. The obstacle avoidance algorithm employed in the simulation is based on and closely resembles the actual e-puck behaviour. This is crucial in order to be able to tell from the simulational results how efficiently multiple robots should be able to complete the path finding task.

2.6 Control Program

The algorithm used in both the simulations and experiments has two basic states. The robot is either searching for food, or searching for the nest. The two states have several subtasks that are ordered by priority and executed in that order. For example, *only* if a robot can distinguish a light trail in front of it, will it try following it.

The two states and their tasks are as follows:

1. Search for food sources (not laying trails, UV-LEDs turned off)
   (a) If the camera registers food (blue), the individual turns 180° and switches to state 2.
   (b) Basic obstacle avoidance, only registers walls or other non-robot objects.
   (c) If the camera registers a green trail (distinguished by contrast in relation to the different camera sectors), follow it. If the sun compass registers that the robot is heading directly to the nest turn 180° .
   (d) Correlated random walk.
2. Search for the nest (laying trails, UV-LEDs turned on)
   (a) If the camera registers the nest (red), the individual turns 180° and switches back to state 1.
   (b) Basic obstacle avoidance, registers non-robots as well as robots and avoids them.
   (c) If the camera registers a green trail (contrast) follow it. If the sun compass registers that the nest is directly behind the individual, turn 180° .
   (d) Follow the sun compass and constantly adjust the trajectory so that the frontmost sensor has the highest value.

3 Results

In order to investigate the attributes and capabilities of our approach, we have run several simulations. They show how a robot swarm of up to twelve robots can achieve better efficiency in finding food and carrying it back to the nest using trails equivalent to the ones on the glowing floor. To measure the accuracy in line-following and of the sun compass on an actual robot, we have carried out test trials using a single robot.

3.1 Foraging Efficiency in Simulation

To measure the efficiency of the setup we count how many times a virtual agent has travelled from food to nest, i.e. how many times it has 'delivered' a unit of food to the nest. Each time an agent completes this task, a global counter variable is increased by one. We have conducted each run with and without pheromone trails and three times in repetition each to eliminate errors. The test runs were run for 2000 time steps. The arena size was fixed at 200×180 patches, while one agent is 7 patches in diameter. The decay properties of the arena floor were set to be consistent with a real-world scenario (see Figure 1). In Figure 4 the layout of the simulated arena can be seen. Figure 5 shows the results of the simulation for 4, 6, 8 and 12 agents in a fixed-size arena. Solid lines indicate efficiency of pheromone-aided robot swarms and dashed lines are from simulation runs without pheromones.

In all of the simulation runs the efficiency is greatly enhanced when utilizing pheromone trails. When four agents where in the arena, pheromone aided robots delivered 8 units, versus only 3 in the test run without pheromones. Six agents archieved 10 with, and 2 without pheromones, eight agents brought 22 vs. 6 and twelve agents 23 vs. 8 units to the nest. This shows that even with a very dense population robots, and the many resulting collisions, a great improvement of efficiency can be archieved using the phosphorescent trails.

3.2 Experiments with the e-Puck Robot Add-Ons

To test our sensor boards, we have conducted the following experiments each with three repetitions and a varying distance between foodsource and nest: The e-puck robot is placed in the middle of the arena and pointed towards the food source. When switched on, it drives straight ahead until its camera recognizes the blue foodsource. It then turns 180° and uses its sun compass sensor to navigate back to the nest. When it has reached the nest it turns 180° and tries to follow its own trail back to the foodsource. This process is repeated for the duration of the experiment. By measuring how often the robot looses the trail we can determine the reliabilty and the limits (in length of trail) of our line-following approach. Note that three components are being evaluated in this process: The camera (for trail-detection), the UV-LED extension (for trail-laying) and the glow-paint floor (how long the trail lasts). In each of our experiments the sun compass always found a nearly straight path back to the nest (see Fig. 6).

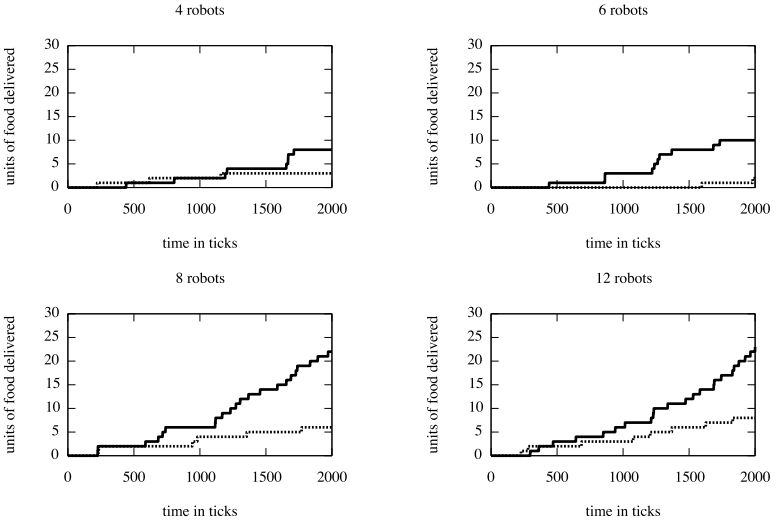


Fig. 5. Efficiency measurements for 4 different simulation runs. Top-left shows 4 robots, top-right 6, bottom-left 8, bottom-right 12. Solid lines are with pheromones turned on, dashed lines represent runs without pheromones. The plots show the sum of units of food delivered to the nest. The efficiency of the agents that could utilize the pheromone trails is greatly increased.

Table 1. Results of experiments carried out with a single robot with varied distances between nest and food

arena size	distance food to nest	food delivered	track lost	time on trail
1m×1m	1.3m	21 units	2 times	82%
1.5m×1.5m	1.6m	10 units	6 times	67%
2m×2m	2.3m	3 units	4 times	49%

We varied the distance of food to the nest for each experiment and repeated them three times each for 15 minutes. Table 1 shows the test results numerically.

With a distance of 1.3 meters from food to nest and a 1m×1m arena, the robot lost track of its trail two times on average. It returned from the foodsource to the nest 21 times and spent 82 percent of the time on the track otherwiese doing a correlated random walk in search of food or using its sun compass to find the nest. Overall line-following reliability was sufficient to keep the glowing trail stable at all times.

Next, the distance was increased to 1.6 meters. The robot lost its trail 6 times on average and brought ten units of food to the nest. It was on the track 67 percent of the time. While the robot lost track of his trail more often, he still succeded in keeping one central trail glowing throughout the experiment.

When the distance was 2.3 meters, most of the time the robot lost its trail halfway to the foodsource, which resulted in inefficient search of the arena using

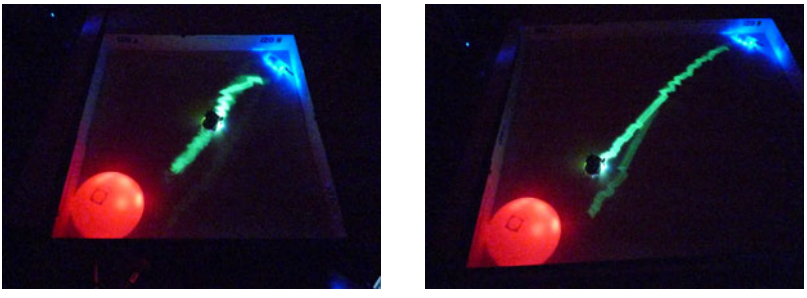


Fig. 6. Left: Photograph showing the trail laid by the robot in a 1m×1m arena and a distance of 1.3m between food (top-right) and nest (bottom-left). The robot is on its way back from the food source pointing towards the nest and is following its own line. **Right**: Same setup in a 1.5m×1.5m arena. Distance between food and nest is 1.6 meters. The robot has lost its track about halfway to the nest and is laying a new one beside the 'old' one.

the correlated random walk. In this case the decay of the phosphorescent floor is too high. Overall it managed to bring back 3 units of food and laid three separate trails on the ground.

4 Discussion and Outlook

In this paper we have introduced an experimental setup to model the foraging behaviour of ants using robots. We have developed solutions to combine a phosphorescent arena floor and robots to lay and follow glowing trails. Additionally we have introduced a novel sensor to mimic the ants capability of using polarization patterns to derive directional information and to navigate back to their nest. Using simulations we have shown that the use of these phosphorescent trails leads to an efficiency increase in collecting units of food and to aid path finding between a food source and the nest. Experimental results show that our newly developed sensors are realiable enough for the robot to navigate to the two spots reliably.

In the future we plan on conducting experiments with more than one robot, in order to transfer the presented simulation into real life. Figure 7 shows a contrived photograph of how this could look like. Collision avoidance between robots is not reliable enough at the moment and needs further investigation.

Our sun compass has proven itself to be very reliable in finding the nest. In combination with the artificial sun, it presents a very cost-effective solution to get directional information inside of a robot arena.

We will explore the capabilities of our sun compass in more detail and introduce more complex behaviours that utilize it, such as remembering the path travelled to the foodsource by the heading in relationship to the sun, also imitating the ants capability to remember their travelled path [6]. In such a scenario the sun could be moved before and during the experiment to show how, just like

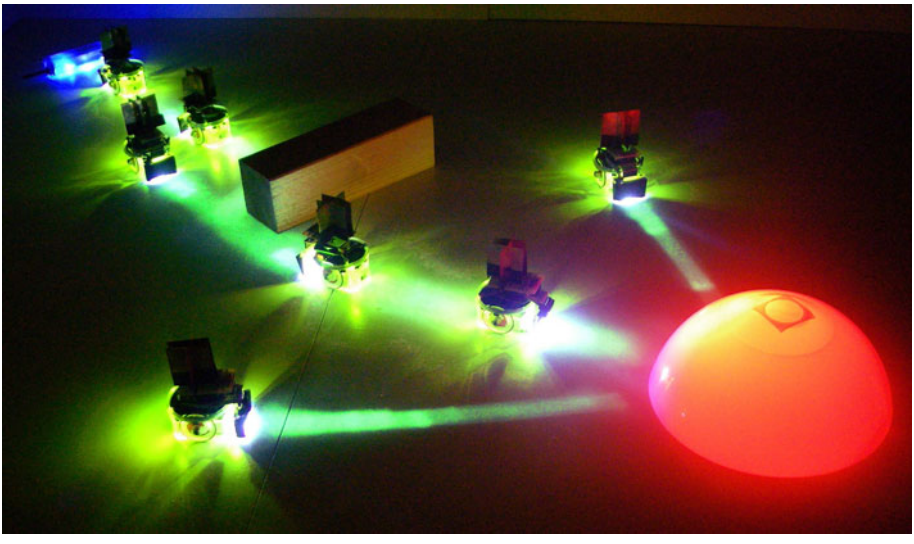


Fig. 7. Contrived photograph of how the glowing floor and our sensors should be used in the future. Two robots leave the nest to search for food, the remaining robots navigate to and from the nest around an obstacle.

in real life, individuals alter their paths accordingly, and steadily adapting to the new position.

Additionally, we will introduce obstacles and multiple food sources in order to explore how the robots optimize the paths in between. The robots could also adopt a tandem behaviour so that they lead each other on pheromone paths to optimize speed and to minimize collisions.

Acknowledgements. This work is supported by the following grants: EU-IST-FET 'SYMBRION', no. 216342; EU-ICT 'REPLICATOR', no. 216240; EU-IST FET 'I-SWARM', no. 507006; FWF (Austrian Science Fund), no. P19478-B16.

References

1. Beckers, R., Deneubourg, J., Goss, S.: Trail laying behaviour during food recruitment in the ant lasius niger (l.). Insectes Sociaux 39(1), 59–72 (1992)
2. Blow, M.: 'stigmergy': Biologically-inspired robotic art. In: Proceedings of the Symposium on Robotics, Mechatronics and Animatronics in the Creative and Entertainment Industries and Arts, pp. 1–8 (2005)
3. Bonani, M., Raemy, X., Pugh, J., Mondana, F., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. In: Proc. of the 9th Conference on Autnomous Robot Systems and Competitions, vol. 1, pp. 59–65 (2009)
4. Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-organization in biological systems. Princeton University Press, Princeton (2001)

5. Dorigo, M., Bonabeau, E., Theraulaz, G.: Ant algorithms and stigmergy. Future Generation Computer Systems 16(9), 851–871 (2000)
6. Fent, K.: Polarized skylight orientation in the desert ant cataglyphis. Journal of Comparative Physiology A: Neuroethology 158(2), 145–150 (1986)
7. Garnier, S., Tache, F., Combe, M., Grimal, A., Theraulaz, G.: Alice in pheromone land: An experimental setup for the study of ant-like robots. In: Swarm Intelligence Symposium, SIS 2007, pp. 37–44. IEEE, Los Alamitos (2007)
8. Müller, M., Wehner, R.: Path integration in desert ants, cataglyphis fortis. Proceedings of the National Academy of Sciences 85, 5287–5290 (1988)
9. Russell, R.: Heat trails as short-lived navigational markers for mobile robots. In: Proceedings of 1997 IEEE International Conference on Robotics and Automation, vol. 4, pp. 3534–3539 (1997)
10. Russell, R.A.: Ant trails – an example for robots to follow? In: Proceedings of 1999 IEEE International Conference on Robotics and Automation, vol. 4, pp. 2698–2703 (1999)
11. Sklar, E.: Netlogo, a multi-agent simulation environment. Artificial Life 13(3), 303–311 (2007)
12. Svennebring, J., Koenig, S.: Building terrain-covering ant robots: A feasibility study. Autonomous Robots 16(3), 313–332 (2004)
13. Turley, J.: Atmel avr brings risc to 8-bit world. Microprocessor Report 11(9) (1997)
14. Wilson, E.O.: Chemical communication in the social insects. Science 149(3688), 1064 (1965)

Automatic Configuration of Multi-Objective ACO Algorithms

Manuel López-Ibáñez and Thomas Stützle

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{manuel.lopez-ibanez,stuetzle}@ulb.ac.be

Abstract. In the last few years a significant number of ant colony optimization (ACO) algorithms have been proposed for tackling multi-objective optimization problems. In this paper, we propose a software framework that allows to instantiate the most prominent multi-objective ACO (MOACO) algorithms. More importantly, the flexibility of this MOACO framework allows the application of automatic algorithm configuration techniques. The experimental results presented in this paper show that such an automatic configuration of MOACO algorithms is highly desirable, given that our automatically configured algorithms clearly outperform the best performing MOACO algorithms that have been proposed in the literature. As far as we are aware, this paper is also the first to apply automatic algorithm configuration techniques to multi-objective stochastic local search algorithms.

1 Introduction

The growing interest on solving optimization problems with respect to multiple, conflicting objectives has led researchers to propose extensions of well-known metaheuristics to tackle them. So far, evolutionary algorithms have received most of the research effort. However, there are also several proposals for applying the ant colony optimization (ACO) metaheuristic [7] to multi-objective combinatorial optimization problems (MCOPs). The majority of these multi-objective ACO (MOACO) algorithms deal with Pareto optimality, that is, they do not make a priori assumptions about the decision maker's preferences.

There are a number of design questions when extending ACO algorithms to MCOPs. First, the meaning of the pheromone information associated to a solution component is unclear in the multi-objective context, because the objective function is multi-dimensional and not scalar, and the output of the algorithm is a nondominated set of solutions and not a single solution. Some MOACO algorithms use several pheromone matrices, each of them associated to a different objective [5,1,10]. If multiple pheromone or heuristic matrices are used, they are aggregated during the solution construction by means of weights [5,10]. However, there are strong differences among MOACO algorithms in the way this aggregation takes place and how many different weights are used. Finally, a further design question is which ants are selected for depositing pheromone. Existing MOACO algorithms select some (or all) nondominated solutions [10],

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 95–106, 2010.

or they select the best solutions with respect to the objective associated to the pheromone matrix that is updated [5,1].

In previous work, we examined alternative design choices of a MOACO algorithm for the bi-objective TSP (bTSP) [16]. Later, we extended this formulation in order to replicate the design of several existing MOACO algorithms from the literature [15]. Our examination of algorithmic components concludes that there are many similarities among the algorithms proposed in the literature. In addition, our empirical results show that, for the bTSP, some algorithmic components achieve clearly superior results in comparison to others.

In this paper, we combine the results of our previous work into an algorithmic framework that may be configured appropriately to reproduce existing MOACO algorithms, and more importantly, it may be configured combining ideas from diverse MOACO algorithms. Thus, this framework facilitates the application of automatic algorithm configuration techniques to configure multi-objective algorithms. However, previous works that aim to produce high performing algorithms by combining a flexible software framework and an automatic tuning tool have dealt so far only with single-objective optimization problems [11]. To the best of our knowledge, there is no previous work on the automatic configuration of stochastic local search (SLS) algorithms for multi-objective optimization problems in terms of Pareto-optimality. Thus, this paper is the first application of such automatic configuration techniques to multi-objective algorithms. We tackle this challenge in a pragmatic way, using unary quality measures, which assign a scalar value to a nondominated set according to some reference criterion, as the optimization goal of an automatic configuration tool. In particular, we configure the proposed framework by applying Iterated F-Race (I/F-Race) [4] using both the hypervolume and the epsilon measures [21].

In summary, the main contributions of this paper are that (*i*) for the first time, we automatically configure MOACO algorithms using unary quality measures, a flexible framework for MOACO algorithms, and a high-performing automatic configuration tool; and that (*ii*) we show that the configurations found automatically are better than configurations implementing several MOACO algorithms proposed in the literature.

2 Experimental Studies on MOACO Algorithms

The available MOACO algorithms differ in a wide diversity of design choices. However, few authors experimentally justify their design choices or compare them to alternatives. Iredi et al. [10] investigated several design alternatives for a multi-colony approach. López-Ibáñez et al. [12] compared different design options for MOACO algorithms on the bi-objective quadratic assignment problem (bQAP). Alaya et al. [1] compared the performance of four MOACO variants for the multi-objective knapsack problem. The review by García-Martínez et al. [8] classifies existing MOACO algorithms according to the usage of one or several pheromone (or heuristic) matrices, and provides a comparison of some of the original algorithms using the bTSP as a case study. The algorithms they tested

differ substantially with respect to underlying ACO parameters, e.g., some of them are based on the classical Ant System, whereas others build upon the typically better performing $\mathcal{MAX}$-$\mathcal{MIN}$ ($\mathcal{MM}$AS) and Ant Colony System (ACS). Therefore, one cannot conclude from the quality of the achieved results on the impact that specific design decisions of each MOACO algorithm have on MOACO performance. In recent research, we first studied alternative design choices for extending ACO algorithms to MCOPs [16] in a systematic manner by keeping other design choices fixed. The design choices studied comprise also those that have been proposed in previous MOACO algorithms. Next, we examined the particular *combinations* of design choices that define existing MOACO algorithms, keeping other factors, such as the underlying ACO algorithm, fixed [15]. The results of these studies showed important differences between various design choices.

3 A Configurable MOACO Framework

Based on our previous work, we identified particular design choices that are clearly superior to others for our case study, the bTSP. At the same time, we realized that some combinations of design choices are promising, but there is no corresponding MOACO algorithm in the literature. Therefore, the next logical step is to propose a configurable MOACO framework that allows us to instantiate several existing algorithms, and variants that have never been proposed before.

We propose the MOACO framework described in Algorithm 1. The framework allows the use of multiple colonies (N^{col}), where each colony constructs solutions independently of others according to its own pheromone information. The colonies cooperate in two ways: (1) by exchanging solutions for updating the pheromone information, and (2) by using a common archive of nondominated solutions for detecting dominated ones. This multi-colony architecture is inspired by the proposal of Iredi et al. [10], which we found more flexible and consistent than other definitions of "colony" [15]. Within each colony, a number of ants construct solutions from the pheromone and heuristic information of their own colony. This pheromone information may be represented either as a single matrix or as multiple matrices that must be aggregated somehow (Aggregation). The same applies to the heuristic information. There are several forms of aggregation, and all of them require the use of a weight. The sequence of weights is determined by the procedure NextWeight. Once there is a unique pheromone and heuristic matrix, a solution is constructed (ConstructSolution), possibly improved by a local search (WeightedLocalSearch) and added to a common archive S^{global}. Once all ants from all colonies have finished constructing solutions, procedure MultiColonyUpdate distributes these solutions among the colonies. From the solutions assigned to each colony, procedure Selection decides which ones will be used for updating the pheromone information (UpdatePheromones). The MOACO algorithm continues until a certain number of iterations or a time limit is reached.

Table 1 describes the configurable settings of the proposed framework. Some settings are only significant for certain values of other settings. For example, the

Algorithm 1. MO-ACO framework

while not stopping criteria met **do**
  for each colony $c \in \{1, \ldots, N^{\text{col}}\}$ **do**
    for each ant $k \in \{1, \ldots, N^{\text{a}}\}$ **do**
      $\boldsymbol{\lambda} := \mathsf{NextWeight}(\Lambda,\ k,\ \mathit{iteration})$
      $\tau := \begin{cases} \mathsf{Aggregation}(\boldsymbol{\lambda}, \{\tau^1, \ldots, \tau^d\}) & \text{if } \texttt{multiple } [\tau] \\ \tau & \text{if } \texttt{single } [\tau] \end{cases}$
      $\eta := \begin{cases} \mathsf{Aggregation}(\boldsymbol{\lambda}, \{\eta^1, \ldots, \eta^d\}) & \text{if } \texttt{multiple } [\eta] \\ \eta & \text{if } \texttt{single } [\eta] \end{cases}$
      $s := \mathsf{ConstructSolutiond}(\tau, \eta)$
      $s := \mathsf{WeightedLocalSearch}(s, \boldsymbol{\lambda})$ // Optional
      $S^{\text{global}} := S^{\text{global}} \cup \{s\}$
    end for
  end for
  for each colony $c \in \{1, \ldots, N^{\text{col}}\}$ **do**
    $S_c := \mathsf{MultiColonyUpdate}(S^{\text{global}})$
    $S_c^{\text{upd}} := \mathsf{Selection}(S_c)$
    $\mathsf{UpdatePheromones}(S_c^{\text{upd}}, N^{\text{upd}})$
  end for
end while
Output: P^{bf}

possible settings of MultiColonyUpdate only make a difference when $N^{\text{col}} > 1$, otherwise all solutions are assigned to the single colony. Both settings, *origin* and *region*, were originally proposed by Iredi et al. [10]. Update by origin assigns each solution from S^{global} to its original colony, whereas update by region divides S^{global} in equal parts among the colonies in such a way that each colony roughly corresponds to one region of the objective space. The alternatives for the Selection component are:

Nondominated solutions. The solutions used for updating the pheromone information are the nondominated solutions in S_c^{upd}. When there are more nondominated solutions than N^{upd}, we apply the truncation mechanism of SPEA2 [20] to select only N^{upd} solutions. It is possible to combine this Selection method and multiple pheromone matrices [10], however, in the case of the bTSP and with $\Delta\tau = 1$, this would result in updating both matrices with the same value, so we do not empirically explore this combination.

Best-of-objective. Selects from the current S_c^{upd}, the N^{upd} best solutions with respect to each objective. In the case of multiple pheromone matrices, each pheromone matrix is updated using the N^{upd} solutions associated to the corresponding objective. Otherwise, the $d \cdot N^{\text{upd}}$ solutions update the single pheromone matrix.

Best-of-objective-per-weight. We keep a list for each weight λ and each objective of the N^{upd} best solutions for each objective generated using λ. In the particular case of $\lambda = 0$, we only keep one list for the first objective,

Table 1. Algorithmic components of the proposed MOACO framework

Component	Domain	Description
N^{col}	$\mathbb{N}^+$	Number of colonies
MultiColonyUpdate	{ origin, region }	How solutions are assigned to colonies for update [10]
N^{upd}	$\mathbb{N}^+$	Max. number of solutions that update each $[\tau]$ matrix
Selection	nondominated solutions, Best-of-objective, Best-of-objective-per-weight	Which solutions are selected for updating the pheromone matrices
$\lvert\Lambda\rvert$	$\mathbb{N}^+$	Number of weights per colony
NextWeight	one weight per iteration, all weights per iteration	How weights are used at each iteration
$[\tau]$	{ single, multiple }	Number of pheromone matrices
$[\eta]$	{ single, multiple }	Number of heuristic matrices
Aggregation	weighted sum, weighted product, random	How weights are used to aggregate different matrices

and we do the same for $\lambda = 1$ and the second objective. When using multiple pheromone matrices, each matrix is updated using only solutions from lists associated to the same objective. This update method is used by the existing mACO-1 and mACO-2 algorithms [1]. It is not clear how this approach should be extended to multiple colonies, since then solutions may be exchanged among colonies with different weights.

The set of weights (Λ) is finite and equally distributed in the interval $[0, 1]$. If there are multiple colonies, Λ is partitioned among the colonies. The options tested for NextWeight are either that all ants in one colony use the same weight at a certain iteration (*one-weight-per-iteration*), or that all weights are used at each iteration (*all-weights-per-iteration*). In the case of one-weight-per-iteration, the weight used by each colony in successive iterations follows an ordered sequence of the elements of Λ, and the order is reversed when the last weight in the sequence is reached. In the case of *all-weights-per-iteration*, when the number of ants N^{a} is larger than the number of weights $|\Lambda|$, then several ants will use the same weight. The aggregation of the pheromone matrices is computed once per weight per iteration.

Lastly, our framework includes three options for Aggregation:

Weighted sum. The two pheromone (or heuristic) matrices are aggregated by a weighted sum: $(1 - \lambda)\tau^1_{ij} + \lambda\tau^2_{ij}$.

Weighted product. The two pheromone (or heuristic) matrices are aggregated by a weighted product: $(\tau^1_{ij})^{(1-\lambda)} \cdot (\tau^2_{ij})^{\lambda}$

Random. At each construction step, an ant selects the first of the two pheromone matrices if $U(0, 1) < 1-\lambda$, where $U(0, 1)$ is a uniform random number, otherwise it selects the other matrix.

Table 2. Taxonomy of MOACO algorithms as instantiations of the proposed framework (d is the number of objectives, N^{a} is the number of ants)

Algorithm	$[\tau]$	$[\eta]$	Aggregation	$\lvert\Lambda\rvert$	Selection
MOAQ [17,8]	1	d	–	d ($\Lambda = \{0, 1\}$ if $d = 2$)	nondominated solutions
P-ACO [5]	d	d	weighted sum	N^{a}	Best-of-objective
MACS [2]	1	d	weighted product	N^{a}	nondominated solutions
BicriterionAnt [10]	d	d	weighted product	N^{a}	nondominated solutions
COMPETants [6]	d	d	weighted sum	$d+1$ ($\Lambda = \{0, 0.5, 1\}$)	Best-of-objective
mACO-1 [1]	d	d	random (τ)/w. sum (η)	$d+1$ ($\Lambda = \{0, 0.5, 1\}$)	Best-of-objective-per-weight
mACO-2 [1]	d	d	weighted sum	$d+1$ ($\Lambda = \{0, 0.5, 1\}$)	Best-of-objective-per-weight
mACO-3 [1]	1	1	–	–	nondominated solutions
mACO-4 [1]	d	1	random (τ)	1 ($\Lambda = \{0.5\}$)	Best-of-objective

Our previous work [15] has examined the design of several multi-objective ACO algorithms from the literature. This previous study has informed the design of the proposed framework. As a result, the framework is flexible enough to allow us to replicate those MOACO algorithms. We provide in Table 2 the configuration of algorithmic components necessary to instantiate several well-known MOACO algorithms.

4 Automatic Configuration of MOACO Framework

Instead of using a trial-and-error approach to identify good instantiations of the proposed MOACO framework, we follow the work of KhudaBukhsh et al. [11] (and earlier work discussed in that paper) in the sense that we use an efficiently implemented, flexible software framework together with an automated algorithm configuration tool for obtaining very high-performing algorithmic variants. However, this is the first time that such an approach is applied in a multi-objective context.

Specifically, we automatically configure our MOACO framework using a new implementation of Iterated F-race (I/F-Race) [4] as the automatic configuration method. As the evaluation criteria used by I/F-Race, we tested two unary measures for evaluating the output of multi-objective algorithms, namely, the hypervolume and the (additive) epsilon measure [21]. The hypervolume is the volume of the objective space weakly dominated by a nondominated set and bounded by a reference point that is strictly dominated by all known points. The larger the hypervolume, the better is the corresponding nondominated set. The additive epsilon measure provides the minimum value that must be subtracted from all objectives of a nondominated set so that it weakly dominates a reference set. This reference set is usually the nondominated set of all known solutions. A smaller epsilon measure is preferable.

Each run of I/F-Race uses a maximum budget of 1 000 experiments and it is repeated five times, for each quality measure, to assess the variability of the automatic configuration process. As training instances, we generated 36 bTSP Random Uniform Euclidean instances for each of $n = \{100, 200, 300\}$ nodes (108

instances in total). Each experiment, that is, each run of the MOACO framework on each instance, is stopped after n CPU-seconds. We provide to I/F-Race appropriate domains of the MOACO framework components. In particular, we use the domains described in Table 1 for the categorical components, with the restrictions described in the text for the Selection component. For the remaining components, we use the following domains: $N^{\text{col}} = \{1, 2, 3, 5, 10\}$, $|\Lambda| = \{2, 6, 12, 24\}$, $N^{\text{upd}} = \{1, 2, 5, 10\}$. We use $\mathcal{MM}$AS and its default parameter settings for the TSP [18] as the underlying ACO algorithm with some exceptions. We use 24 ants per colony, which is a value close to the one used in the MOACO literature for the bTSP [8] and it allows us to divide the ants exactly among several values of $|\Lambda|$. We also use $\Delta\tau = 1$ for the amount of pheromone deposited by an ant, and the evaporation rate is set to $\rho = 0.05$. As for the optional local search, we use a 2-opt algorithm that exploits candidate lists of length 20. More details on the local search are given in our previous work [16].

We perform runs of I/F-Race with and without local search. The rationale for separating both analyses is that, on the one side, it is clear that MOACO algorithms with local search by far outperform those without local search for the bTSP [16]; on the other side, given the large amount of work on MOACO algorithms without the usage of local search [8], it is interesting to examine whether the automated configuration of the MOACO framework may obtain competitive results in comparison to those algorithms.

We have implemented the MOACO framework in `C` using ACOTSP (`http://www.aco-metaheuristic.org/aco-code`) as the underlying ACO package, and compiled it with `gcc`, version 3.4. All experiments reported in the following are carried out on AMD Opteron 2216 dual-core 2.4 GHz processors with 2 MB L2-Cache under Rocks Cluster GNU/Linux. The implementation is sequential and experiments run on a single core.

The results of the automatic configuration with respect to hypervolume and epsilon measures are very similar. In particular, the configurations obtained when maximizing the hypervolume are very consistent. In all five runs of I/F-Race, the best configuration uses multiple colonies (varying among three, five and ten colonies), selection by best-of-objective, one weight per iteration, multiple pheromone and heuristic matrices, and aggregation by weighted product. The differences are in the multi-colony update, I/F-Race chooses three times *update by region*, and two times *by origin*; in the number of solutions used for update, which varies among $N^{\text{upd}} = \{1, 2, 5, 10\}$; and in the number of weights, which varies among $|\Lambda| = \{2, 6, 12\}$. The results of I/F-Race when optimizing for the epsilon measure are more varied. The settings that were common to all five runs are multiple colonies (either five or ten), multiple heuristic matrices, and aggregation by weighted product. One run of I/F-Race chose the use of nondominated solutions for the Selection component; surprisingly this was selected together with $N^{\text{upd}} = 1$, which effectively means that the chosen solution would be the best for one of the objectives. That same configuration uses a single pheromone matrix and multiple heuristic matrices. The other four runs of I/F-Race with epsilon measure follow the results obtained with the hypervolume measure and

produce configurations using selection by best-of-objective and multiple pheromone matrices. For the other parameters, the automatically configured settings vary among $N^{\text{upd}} = \{1, 2, 5, 10\}$ and $|\Lambda| = \{6, 12, 24\}$.

We repeat the automatic configuration with 2-opt local search enabled. In this case, the resulting configurations are even more varied. This probably indicates that once local search is enabled, the particular settings of the other parameters are less important. Focusing on the common settings, all resulting configurations use multiple colonies (nine times ten colonies, and once five colonies). Moreover, in most configurations found using the hypervolume measure there are multiple pheromone and heuristic matrices and Selection uses best-of-objective, whereas in those found using the epsilon measure is more common the use of a single pheromone matrix and Selection uses nondominated solutions.

5 Comparison with Existing MOACO Algorithms

We now compare the automatically configured MOACO algorithms with the results obtained by existing MOACO algorithms from the literature. A recent comparison of MOACO algorithms for the bTSP identified BicriterionAnt as the best-performing overall [15], so we will use this algorithm for comparison.

First, we choose the best automatically configured MOACO algorithm by performing 15 runs of each configuration returned by I/F-Race on each of the 108 tuning instances. From these results, we choose the configuration (AutoMOACO) with the largest mean hypervolume and smallest mean epsilon measure. In this case, both measures agree on the best configuration: ten colonies, multi-colony update by origin, selection of best-of-objective, one-weight-per-iteration, $N^{\text{upd}} = 2$, $|\Lambda| = 12$, aggregation by weighted product, and multiple pheromone and heuristic matrices.

We compare BicriterionAnt and AutoMOACO on three instances (`kroAB100`, `kroAB200`, and `euclidAB300`), different from the ones used for tuning, and taken from Luis Paquete's webpage (`http://eden.dei.uc.pt/~paquete/tsp`). As before, we perform 15 runs of each algorithm on each test instance and each run has a time limit of n seconds. For the comparison, instead of relying on the same quality measures used for tuning, we examine the differences between the algorithms' empirical attainment functions (EAFs) [13,14]. The EAF estimates from several independent runs the probability of a multi-objective algorithm obtaining or dominating a certain point in the objective space [9]. A plot of the EAF differences between two algorithms gives on each side the differences in favour of either algorithm. Larger differences are indicated in darker color. Large differences indicate one algorithm has a higher probability of dominating a certain region of the objective space than the other algorithm. Figure 1 gives two of such plots, comparing BicriterionAnt and AutoMOACO on the two largest test instances. In all plots, there are only differences in favor of AutoMOACO (dark points occur only on the right plots), which indicates that AutoMOACO completely outperforms BicriterionAnt.

In a second comparison, we consider the MOACO variants with local search. We repeat the same procedure described above for choosing the best configuration

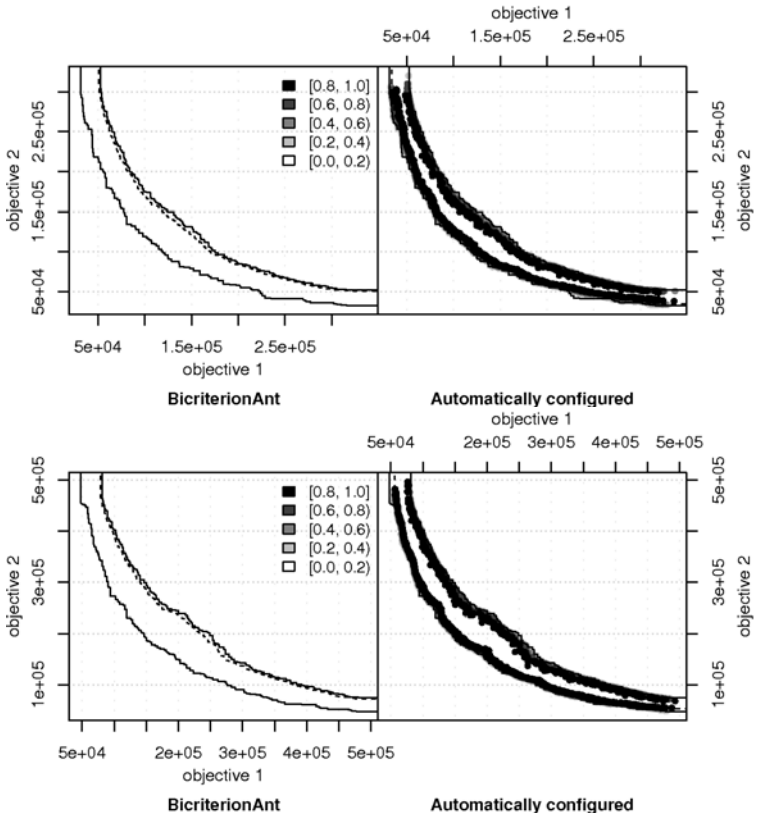


Fig. 1. The plots show the differences between the EAF of BicriterionAnt vs. AutoMOACO on instances `kroAB200` (top) and `euclidAB300` (bottom). The magnitude of the differences is encoded in grey-scale.

from the ten runs of I/F-Race (five runs with each quality measure), that is, we perform 15 independent runs of each configuration on the 108 tuning instances, and choose the configuration with the best mean value of the quality measures. Both measures again agree on the best automatically configured MOACO using local search (AutoMOACO+ls): ten colonies, multi-colony update by region, one-weight-per-iteration selection of nondominated solutions, $N^{\text{upd}} = 1$, $|\Lambda| = 12$ and a single pheromone matrix.

We compare AutoMOACO+ls, on the three test instances mentioned above, to the algorithm suggested by our earlier studies on MOACO components [16]. Based on that paper, we select a configuration with weighted sum aggregation (given that weighted product aggregation was not studied there), one colony, multiple pheromone and heuristic matrices, and one weight per iteration. Although it is difficult to appreciate the EAF differences in the graphical plots, as illustrated by Fig. 2, the differences are strongly in favor of AutoMOACO+ls

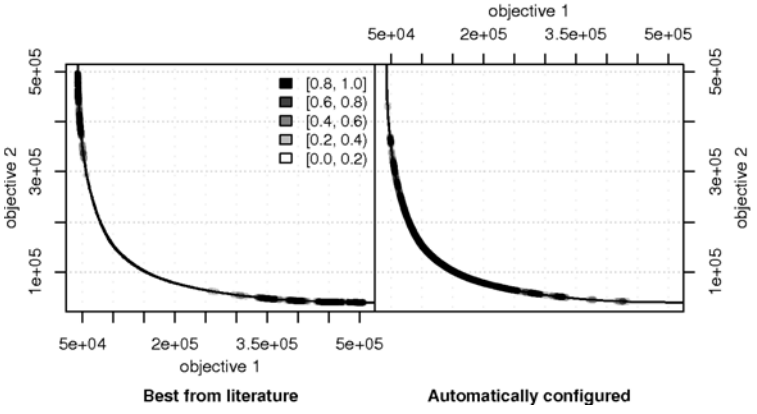


Fig. 2. The plot shows the differences between the EAFs of two MOACO algorithms using local search: best configuration from the literature [16] vs. AutoMOACO+ls. The instance is `euclidAB300`. The magnitude of the differences is encoded in grey-scale.

and there are only small differences in favor of the literature-best version in very few points of the objective space. To confirm these differences, we apply a non-parametric Wilcoxon rank-sum test on the hypervolume values obtained by AutoMOACO+ls and the literature-best version. The test rejects the hypothesis that both algorithms reach similar hypervolume at a significant level of 0.05. Since AutoMOACO+ls obtains a lower mean hypervolume than the literature-best version, we conclude that there is a statistically significant difference in favor of AutoMOACO+ls. We repeat this procedure for each instance and for each quality measure (hypervolume and epsilon). In all cases the statistical tests are in favor of AutoMOACO+ls, which confirms that AutoMOACO+ls outperforms the best MOACO algorithm with local search from the literature in these bTSP instances.

6 Conclusions

In this paper, we propose a flexible, configurable ACO framework for multi-objective problems in terms of Pareto optimality. This framework is the result of our review and synthesis effort of different MOACO algorithms from the literature. The framework may be configured to instantiate those existing algorithms, and others never examined before in the literature. We demonstrate the potential of this approach by automatically configuring our framework for the bTSP and comparing it with a state-of-the-art MOACO algorithm for this problem.

We automatically configure our MOACO framework by applying I/F-Race using unary quality measures to evaluate the nondominated sets returned by multi-objective optimization algorithms. We repeated the automatic configuration with two different unary measures, the hypervolume and the epsilon measure, and we found that there are some small differences in the quality of the

algorithmic configurations obtained. Although it may be possible to use both measures at the same time, such approach would require defining a consensus method in case both measures contradict each other. The proposed approach is simple, pragmatic and effective. However, further work is underway to investigate the use of different unary and binary measures, and how to adequately combine several quality measures at the same time.

The result of the automatic configuration is a MOACO algorithm that mixes features from existing algorithms and is able to outperform them in the bTSP. This result is a further confirmation that automatic configuration of a flexible framework may surpass human-designed algorithms [11]. The main contribution of this paper, however, is not the fine-tuned algorithm, but rather the MOACO framework itself and the method proposed here for automatically fine-tuning it. First, the proposed framework can be applied to any problem for which the reviewed MOACO algorithms have been applied so far. Second, the proposed configuration method may be used to fine-tune the framework to the particular problem and, possibly, improve over existing results. Moreover, the proposed configuration method can be applied to any multi-objective optimization algorithm, which so far are either not fine-tuned at all or by trial-and-error. Future work would extend the MOACO framework to include more diverse algorithmic components, and incorporate ideas from multi-objective evolutionary algorithms.

Acknowledgments. This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium. Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is a Research Associate. The authors also acknowledge support from the FRFC project "*Méthodes de recherche hybrides pour la résolution de problèmes complexes*".

References

1. Alaya, I., Solnon, C., Ghédira, K.: Ant colony optimization for multi-objective optimization problems. In: 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), vol. 1, pp. 450–457. IEEE Computer Society Press, Los Alamitos (2007)
2. Barán, B., Schaerer, M.: A multiobjective ant colony system for vehicle routing problem with time windows. In: Proceedings of the Twenty first IASTED International Conference on Applied Informatics, Insbruck, Austria, pp. 97–102 (2003)
3. Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuß, M. (eds.): Experimental Methods for the Analysis of Optimization Algorithms. Springer, Heidelberg (2010)
4. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-race and iterated F-race: An overview. In: Bartz-Beielstein, et al [3] (to appear)
5. Doerner, K.F., Gutjahr, W.J., Hartl, R.F., Strauss, C., Stummer, C.: Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. Annals of Operations Research 131, 79–99 (2004)
6. Doerner, K.F., Hartl, R.F., Reimann, M.: Are CompetAnts more competent for problem solving? The case of a multiple objective transportation problem. Central European Journal for Operations Research and Economics 11(2), 115–141 (2003)

7. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
8. García-Martínez, C., Cordón, O., Herrera, F.: A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. European Journal of Operational Research 180(1), 116–148 (2007)
9. Grunert da Fonseca, V., Fonseca, C.M., Hall, A.O.: Inferential performance assessment of stochastic optimisers and the attainment function. In: Zitzler, et al [19], pp. 213–225
10. Iredi, S., Merkle, D., Middendorf, M.: Bi-criterion optimization with multi colony ant algorithms. In: Zitzler, et al [19], pp. 359–372
11. KhudaBukhsh, A.R., Xu, L., Hoos, H.H., Leyton-Brown, K.: SATenstein: Automatically building local search SAT solvers from components. In: Proc. of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 517–524 (2009)
12. López-Ibáñez, M., Paquete, L., Stützle, T.: On the design of ACO for the biobjective quadratic assignment problem. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 214–225. Springer, Heidelberg (2004)
13. López-Ibáñez, M., Paquete, L., Stützle, T.: Hybrid population-based algorithms for the bi-objective quadratic assignment problem. Journal of Mathematical Modelling and Algorithms 5(1), 111–137 (2006)
14. López-Ibáñez, M., Paquete, L., Stützle, T.: Exploratory analysis of stochastic local search algorithms in biobjective optimization. In: Bartz-Beielstein, et al [3], 209–233
15. López-Ibáñez, M., Stützle, T.: The impact of design choices of multi-objective ant colony optimization algorithms on performance: An experimental study on the biobjective TSP. In: GECCO 2010, pp. 71–78. ACM Press, New York (2010)
16. López-Ibáñez, M., Stützle, T.: An analysis of algorithmic components for multiobjective ant colony optimization: A case study on the biobjective TSP. In: Collet, P., Legrand, P. (eds.) EA 2009. LNCS, vol. 5975, pp. 134–145. Springer, Heidelberg (2010)
17. Mariano, C.E., Morales, E.: MOAQ: An Ant-Q algorithm for multiple objective optimization problems. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), vol. 1, pp. 894–901. Morgan Kaufmann Publishers, San Francisco (1999)
18. Stützle, T., Hoos, H.H.: $\mathcal{MAX}$-$\mathcal{MIN}$. Future Generation Computer Systems 16(8), 889–914 (2000)
19. Zitzler, E., Deb, K., Thiele, L., Coello, C.A., Corne, D. (eds.): EMO 2001. LNCS, vol. 1993. Springer, Heidelberg (2001)
20. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K., et al. (eds.) Proceedings of EUROGEN 2001, International Center for Numerical Methods in Engineering (CIMNE), pp. 95–100 (2002)
21. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)

Autonomous Morphogenesis in Self-assembling Robots Using IR-Based Sensing and Local Communications

Wenguo Liu and Alan F.T. Winfield

Bristol Robotics Laboratory, University of the West of England, Bristol, UK
Wenguo.Liu@brl.ac.uk, Alan.Winfield@uwe.ac.uk

Abstract. This paper presents a simple decentralised morphology control mechanism for a swarm of self-assembling robots. Each robot in the system is fully autonomous and controlled using a behaviour-based approach with only infrared-based local sensing and communications. A graph-based recruitment strategy is proposed to guide the growth of 2D planar organisms, and local communications are used to self-organise the behaviours of robots during the morphogenesis process. The effectiveness of the approach has been verified, in simulation, for a diverse set of target structures.

1 Introduction

The EU-funded project SYMBRION (http://www.symbrion.eu) is aiming to develop a super-large-scale swarm of robots which is able to autonomously assemble to form 3D symbiotic organisms to perform complex tasks. The idea is to combine the advantages of swarm and self-reconfigurable robotics systems to investigate and develop novel principles of evolution and adaptation for robotic organisms from bio-inspired and evolutionary perspectives [5]. Unlike modular self-reconfigurable robotic systems such as PolyBot G3 [13], CONRO [8], M-TRAN III [7] and SuperBot [9], in SYMBRION individual robots are independently mobile and will be able to autonomously aggregate and dock with each other. The robots will initially form a 2D planar organism. Once the robots in the 2D planar organism have assumed the correct functionality, according to their position in the organism, the organism will lift itself from 2D planar configuration to 3D configuration and, with respect to locomotion, will function as a macroscopic whole. The aggregated organism will also be able to disassemble and reassemble into different morphologies to fit the requirements of the task.

The morphologies of the organism that the robots can self-assemble into must be constrained by the specific hardware design of the individual robots. With only limited sensory capabilities, it is a challenge to coordinate the behaviours of a large number of robots in a decentralised manner in order that the robots can form some desired structures. Various morphology control mechanisms have been proposed for controlling different modular robotic systems in recent years. Støy [11] has evaluated a gradient-based approach to control the self-reconfiguration

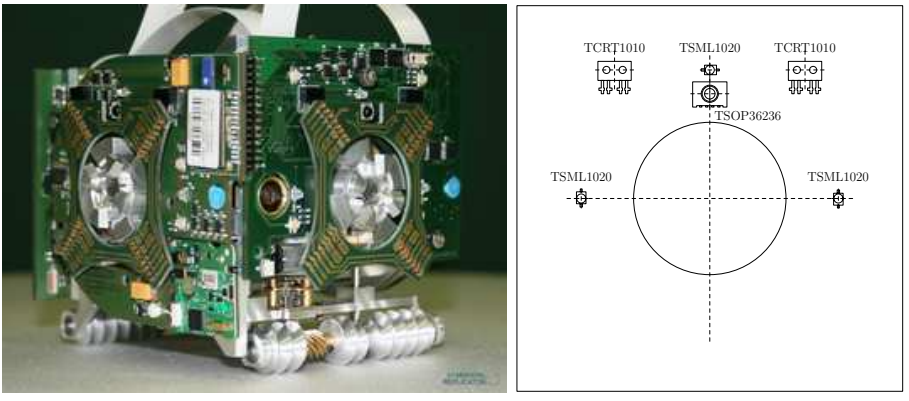
M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 107–118, 2010.

of cubic units in simulation, where the desired configuration is grown from an initial seed module and guided by the gradient in the system using local communication. Guo et al [3] proposed a distributed gene regulatory network (GRN) based algorithm for multi-robot construction, in which the global shape information is embedded into the GRN dynamics directly and the local interaction among the robots is represented by the diffusion terms; they showed, in simulation, that different pre-defined simple shapes can be formed. Also tested in simulation, Grushin and Reggia [2] developed an automated rule generation procedure that allows structures to successfully self-assemble in an environment with constrained, continuous motion. Shen et al. [10] applied a bio-inspired hormone-based control mechanism for the CONRO robots to coordinate motions and perform reconfiguration. The hormone is used to trigger different actions in different modules and is modelled as special messages transferred among these modules via limited local communication. Apart from controlling the morphologies of lattice type or chain type robots, Christensen et al. have proposed a simple language, SWARMMORPH-script, for arbitrary morphology generation for self-assembling robots [1], where each robot is fully autonomous. The morphologies are pre-specified as sets of rules stored in scripts which can be communicated and subsequently executed on the newly connected robot. Their morphology control algorithm has been demonstrated using a group of *s-bot* robots in a 2D environment. This study also needs to consider the morphology control problem for a swarm of autonomous mobile robots. However, this paper focuses on how specific structures can be formed based on the existing sensing and communication capabilities of the SYMBRION robot.

2 SYMBRION Robots and Their Docking Sensors

Figure 1(a) shows the first generation of a SYMBRION robot. It has a cubic shape sized 8cm x 8cm x 8cm. The robot can move omnidirectionally in a 2D planar environment using two screwdrive type wheels, and bend 90 degrees along the common axis of two opposite docking units using a hinge drive, which is in parallel with the wheel axis. A rich set of sensors are proposed to be installed in the robot for environmental perception, locomotion and internal state monitoring purposes, see [4] for a full list. Four mechanical docking units, one on each vertical side, are installed on the robot to allow stable physical connections between robots. In addition, electrical contacts next to the docking units can be coupled automatically to provide inter-robot communication and power sharing busses between two connected robots. The docking units can handle misalignment in horizontal and vertical directions as well as rotation within certain ranges.

To achieve autonomous docking in a 2D planar environment, specific infrared (IR)-based sensing – including proximity detection and docking alignment detection – and local communications circuits have been developed for the SYMBRION robot, see [6]. Each robot is endowed with 8 proximity sensors, 8 docking alignment sensors and 4 channel local communications for autonomous docking, the maximum detection range for each function is about 15cm, 25cm and 150cm



(a) (b)

Fig. 1. a) The first generation prototype of a SYMBRION robot and, b) the placement of the IR sensors on each vertical side PCB

respectively. These sensors have the same placement on each side PCB of the robot, as shown in Figure 1(b). More specifically, two IR sensors (TCRT1010) have been placed symmetrically above and on either side of the docking unit (marked with a circle); one IR LED (TSML1020) is placed directly above the docking unit, while the other two LEDS are located on either side of the docking unit. These LEDs are used to emit different frequency signals for obstacle detection, docking alignment and communication. The IR sensors work for both obstacle detection and docking alignment detection. As for communications, one IR remote control receiver (TSOP36236) is placed next to the IR LED on each side PCB. Note that the 4 channels of local communication can work simultaneously. By default they are all in "listening" mode; whenever one robot is broadcasting messages, another robot within range will receive the message with one or two adjacent channels, which provide the robot with an approximation of the direction of the signalling robot.

3 Robot Controller Design

Consider one scenario for a swarm of SYMBRION robots: initially some robots are randomly deployed in the environment to perform a specified task, for example, searching for a power socket at a certain height that cannot be reached by a single robot. In this phase all robots must rely on their own sensing and computation and are in so-called *Swarm Mode*. The robot that senses the power socket first will become the seed robot and hence initiate the process of self-assembly for an organism that might be able to reach the power socket. All robots will have the same knowledge about what kind of structures they can self-assemble into, however, the actual structure to be instantiated must be chosen by the seed robot. The seed robot then changes its state to *Organism Mode*

and broadcasts signals to recruit other robots for docking. It will send a message to the next robot that successfully docks with it, containing the identity of the structure that is being assembled. The same process is repeated by the newly docked robot until the specified structure is formed. Thereafter, the robots in the organism must determine collectively whether the current structure is suitable for the task, e.g. to reach the power socket, or not. If not, a new shape must be selected; all or some of the robots must disconnect from the organism and a new cycle of self-assembly started until the organism can achieve its goal. Note that questions such as how the seed robot chooses the best organism shape and how the organism determines whether or not it can achieve its goal are beyond the scope of this paper. A behaviour-based approach is adopted for the design of the morphogenesis controller as described in the following sections.

3.1 A Finite State Machine

Figure 2 shows the finite state machine (FSM) for the morphogenesis controller. Depending on the physical connection status of the robot, the 7 states in the FSM can be categorised into two blocks as marked with dashed lines in Figure 2 – *Swarm Mode* and *Organism Mode*. Switching between these two modes occurs whenever a robot either docks with or undocks from another robot in the organism. For the robots in *Organism Mode*, the default state is *InOrganism*; this may change to state *Recruitment* or *Disassembly* during the self-assembly process and transitions are determined by the morphogenesis strategy applied by robots. Once robots are in state *Recruitment*, they will flash one of their IR LEDs – the docking beacon – to attract other robots in *Swarm Mode* to dock. For the robots in *Swarm Mode*, when a docking beacon is sensed they will move

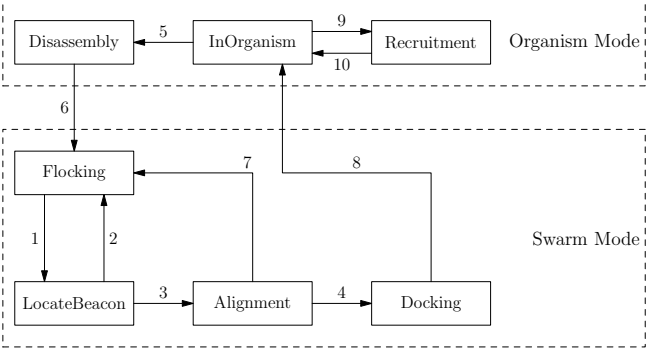


Fig. 2. Robot finite state machine (FSM) for autonomous morphogenesis controller. Conditions causing state transitions: 1 – docking message received; 2 – collision, or no docking message received; 3 – docking beacon signals detected; 4 – aligned and ready to dock; 5 – disassembly required; 6 – undocking completed; 7 – expelling message received, or docking signals lost; 8 – docking completed; 9 – recruitment required; 10 – recruitment completed.

towards it and try to dock to the recruiting robot accordingly; here transitions from one state to another are triggered by the combination of IR sensing and communication. Note that *Flocking* is a place holder for all other swarm mode behaviours, not associated with self-assembly or disassembly.

3.2 Local Communication

Local communication is used to self-organise the behaviour of the robots and resolve competition when self-assembly is in progress. Some simple communication protocols are implemented here, with consideration to the capability of the robots' IR communications. Five fixed message tokens, each of 1-Byte length, are broadcast by the robots when communication is required, as follows:

MSG-Recruitment is to indicate that a recruitment process has started. The message is broadcast and repeated by the robots in state *Recruitment*. It is used by other robots to locate the direction of a recruitment robot in longer range with less accuracy.

MSG-InRange is transmitted by the robot in state *LocateBeacon* when it detects beacon signals (transmitted by one of the IR LEDs of a recruitment robot). The message is used to inform the recruitment robot to stop transmitting MSG-Recruitment messages.

MSG-Expelling is broadcast by the robot in state *Alignment* to expel other competitors in order to make more room for docking alignment and thus reduce interference.

MSG-DockingReady is sent by the robot in state *Docking* when its docking unit is fully in position to the recruitment robot. It is used to inform the recruitment robot to stop emitting beacon signals and start to lock the docking units.

MSG-NewRobotAttached is initially transmitted by the recruitment robot when a new robot is docked. The message is then propagated by every docked neighbour robot in the organism. It is used to trigger the transitions between states *InOrganism*, *Disassembly* and *Recruitment* along with the morphogenesis strategies explained later.

MSG-UnDocked is sent by the robot in state *Disassembly* when the undocking procedure is fully completed. The robot which was previously docked will receive this message.

Apart from these fixed content message tokens, the robot in state *Recruitment* also needs to send a message to the newly docked robot which includes the current number of robots in the organism and the information of the structure the robots are trying to grow. This message is essential to the implementation of the recruitment strategies discussed later. Note that when transmitting messages, only one or two specific communication channels are used. Since the IR signals may be occluded and have a certain transmission angle and range, the number of candidate receivers is limited, as we would expect.

3.3 Behaviours

The behaviours of each state of the FSM are defined as follows:

InOrganism Robot remains static in the organism while monitoring the 4 communication channels. When a **MSG-NewRobotAttached** message is received from one of the channels, it checks whether it needs to switch to state *Recruitment* or *Disassembly* following certain rules. Then it sends the **MSG-NewRobotAttached** messages to other docked neighbour robots, excluding the one it received the message from.

Recruitment Robot chooses one side, based on the recruitment strategy, from which to emit beacon signals and **MSG-Recruitment** messages at the same time. Once it detects a **MSG-InRange** message, it stops transmitting **MSG-Recruitment** to avoid attracting too many robots. The robot performs a mechanism docking lock when the **MSG-DockingReady** message is received. It then moves to state *InOrganism* and send **MSG-NewRobotDocked** messages to all connected robots.

Disassembling Robot executes an action sequence to undock from the organism if only one of its docking units is connected. It then sends a **MSG-UnDocked** message to the robot previously connected and moves to state *Flocking*. If more than one docking units are connected, it continues to wait.

Flocking Robot wanders in the environment and searches for docking beacons. It avoids obstacles and other robots. When **MSG-Recruitment** messages are received it moves to state *LocateBeacon*.

LocateBeacon Robot approximately locates the beacon using 4 IR communication channels and moves in the direction of the beacon signals. If no **MSG-Recruitment** messages are received, or obstacles are detected, it transfers back to state *Flocking*. If beacon signals are detected, it sends a **MSG-InRange** message and then moves to state *Alignment*.

Alignment Robot adjusts its headings and tries to minimise the misalignment of two docking units. It transmits **MSG-Expelling** messages repeatedly to expel competitors. However, if it detects **MSG-Expelling** messages from other robots, it exits to state *Flocking*. Once two docking units are aligned and close enough (based on readings from the beacon detection sensors and proximity sensors), it transmits a **MSG-DockingReady** message and moves to state *Docking*.

Docking Robot performs a mechanical docking procedure to physically connect to the organism. It moves to state *InOrganism* upon completion.

4 Recruitment Strategies

Although all robots in the swarm have common knowledge of the structures of the organism that they may construct, to grow a specific shape from one seed robot the right strategy is required. In other words, the robots in the partially assembled organism must determine the location and the timing at which a new robot needs to be recruited and connected. As IR signals are used for recruitment

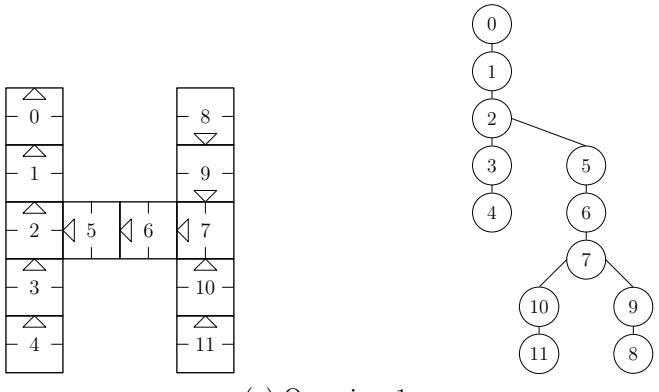
Fig. 3. A robot and its graphical node counterpart. The hinge joint is on the left-right axis.

and docking alignment, interference may arise if more than one light source are actively emitting IR signals for this purpose, at the same time. To overcome this problem, a simple solution – although perhaps not the most efficient one – is to allow only one of the robots in the organism to transmit docking beacon signals and recruitment messages, i.e. to be in state *Recruitment*, at any one time. Since a SYMBRION robot has four docking faces, it is also important to know onto which face a new robot must dock. These problems are referred to as recruitment strategies in this paper.

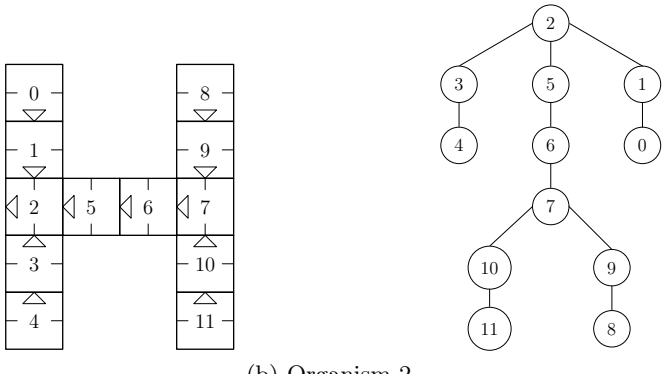
Before we can address the recruitment strategy a common representation of the pre-defined organism structures, to be stored on each robot, must be defined. During an autonomous docking process a recruitment robot is normally static while emitting the docking beacon signals. Although each robot has four side docking mechanisms named front, left, back and right, the locomotion capability of a single robot dictates that robots will use their front side only to dock onto the recruiting robot. Therefore, for any connection between two docking units in the organism, one and only one front side docking unit must be present. If each robot in the organism is treated as a node in a tree data structure where the "parent", "lchild", "mchild" and "rchild" of the node represent the front, left, back and right side of a robot respectively, as shown in Figure 3, then the whole organism in a 2D planar environment can be represented as a tree data structure in which each edge denotes a physical docked connection between two robots. Figure 4(b)(c) show two organisms and their corresponding tree data structure representations. Note that each robot in the organism has been identified with a unique ID number. Although these two organisms have very similar 2D structures, because of the orientation of the hinge driver of the robots (marked with two line segments from the left and right sides of a robot in Figure 4), they have different 3D locomotion capabilities. Clearly, the start point for self-assembly of an organism, i.e. the seed robot, cannot be arbitrarily chosen. It must be the root node of its corresponding tree representation. The order in which robots attach to the organism can be retrieved by a pre-order walk of its tree representation. Assume the children of a node are visited in the order "mchild – lchild – rchild", then for organism 1 shown in Figure 4(b), the robots can be recruited to the organism in the order of list {0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 9, 8}, named *sortedNodeList*, where the first robot, No. 0, will act as a seed robot. Other robots in the list are recruited by their parent node one by one. The order that the robots move into the *Recruitment* state is in fact the order of the

parent nodes of each node in the pre-order walk node list, i.e. {0, 1, 2, 3, 2, 5, 6, 7, 10, 7, 9} for organism 1. The recruitment side of each recruitment robot can also be easily retrieved from the tree representation. If we introduce an ordered pair "(Robot-ID, Recruitment-Side)", then to grow organism 1, the order that the robots move to state *Recruitment* and their corresponding recruitment sides can be expressed as list {(0, 0), (1, 0), (2, 0), (3, 0), (2, 2), (5, 0), (6, 0), (7, 1), (10, 0), (7, 2), (9, 0)}, named *recruitmentNodeList*, where number 0, 1, 2 in the second element of each pair denote the Back, Left and Right side of a robot respectively. Similarly, for organism 2, *sortedNodeList* = {2, 5, 6, 7, 10, 11, 9, 8, 3, 4, 1, 0}, and *recruitmentNodeList* = {(2, 0), (5, 0), (6, 0), (7, 1), (10, 0), (7, 2), (9, 0), (2, 1), (3, 0), (2, 2), (1, 0)}.

Together with the local communication protocols, a pair of *sortedNodeList* and *recruitmentNodeList*, stored in each robot, give sufficient information for the swarm to self-assemble to a specific 2D organism. Take organism 2 as an example, the recruitment strategies are described as follows: the seed robot first retrieves its ID from the *sortedNodeList* and the recruitment side from the



(a) Organism 1



(b) Organism 2

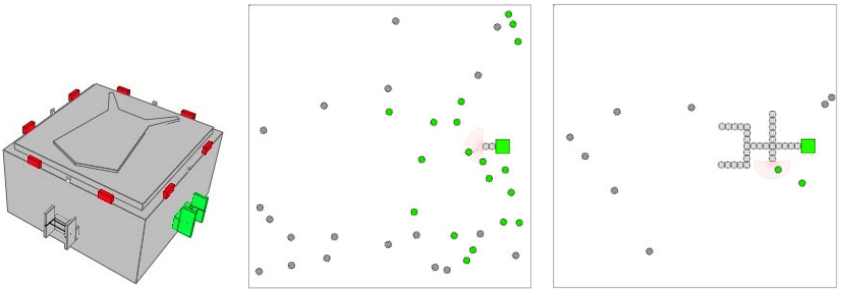
Fig. 4. Graphical representation of organism structures

recruitmentNodeList, where ID = 2, side = 0 (Back). It then starts to emit **MSG-recruitment** messages and docking beacon signals to recruit other robots. When a new robot is docked to its Back side, it sends a message to this robot with the index of the organism and how many robots are in the organism; here index is 2 (corresponding to organism 2) and the number of robots in the organism is 2. The newly docked robot then retrieves its ID from the corresponding *recruitmentNodeList*, here 5 as it knows it is the second robot in the organism. These two robots then move to state *InOrganism*, where they compare their IDs with the ID of the second pair element in the *recruitmentNodeList*. Since it is "(5, 0)", the robot in the organism with ID "5" moves into state *Recruitment* with side 0 (Back) to attract another robot. Similarly, the newly docked robot will receive the index of the organism and the current number of robots in the organism from robot "5", it is then assigned an ID of "6". Meanwhile, robot "5" will transmit a **MSG-NewRobotDocked** message via its Front side. Robot "2" receives this message and will increment its internal variable *numRobotsInOrganism* by 1, now 3. Next, robot "6" in state *InOrganism* will be matched as the recruiting robot from the *recruitmentNodeList*. The process continues until all robots' *numRobotsInOrganism* is equal to the size of the *sortedNodeList*.

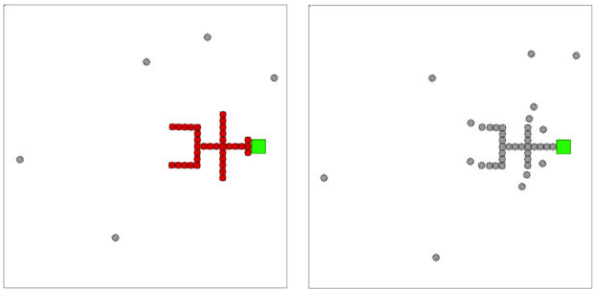
5 Results and Discussion

At the time of writing the SYMBRION robot is still under development and not enough real robot platforms are available for testing the morphogenesis approach presented in this paper. Thus a simulated model of the SYMBRION robot has been implemented in the popular simulation tool Stage [12]. As shown in Figure 5(a), the robot model in Stage has the same size as the SYMBRION robot. For each robot in Stage, the IR-based sensing and communications approach described in [6] is accurately simulated and calibrated with data measured from real sensors. Each robot can move in the arena using two differentially driven wheels (not shown in Figure 5(a)). Four simplified docking units on each vertical face of the robot simulate mechanical docking. As the morphogenesis approach discussed in the paper takes place exclusively in a 2D environment, neither the hinge driver of the robot nor the physics needs to be simulated.

Simulation experiments are carried out within an 8m x 8m bounded arena. 40 robots are deployed, each running the same controller described in previous sections. Figure 5(b)-(e) show screenshots from the Stage simulation in which the robots are self-assembling into a complex 2D shape with 4-way and 3-way joints, and right angles. To trigger the start of the morphogenesis process a large box acting as a "power socket", emitting IR signals which can be detected by the docking sensors of a robot, is placed in the arena. The first robot that finds the box becomes the seed robot and docks with the box. It then chooses, at random, one organism shape from its set of pre-defined structures and executes the recruitment strategy described above to recruit other robots and hence initiate the new structure. To further test the controller, once the organism has completely formed (Fig. 5(d)), all robots in the organism are switched to state *Disassembly*.

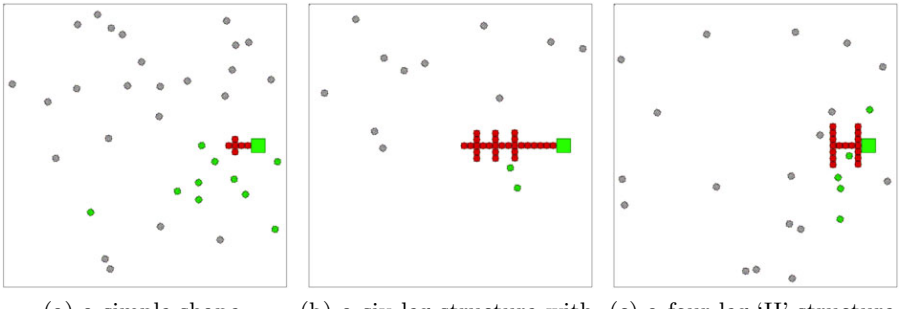


(a) robot model (b) 45m 41s 700msec (c) 1h 18m 24s 500msec



(d) 1h 32m 47s 500msec (e) 1h 33m 01s 500msec

Fig. 5. Screenshots from simulation, the first robot is attached to the large box at time 42m 52s 600msec. The organism is completed at time 1h 32m 30s 600msec.



(a) a simple shape (b) a six leg structure with tail (c) a four leg 'H' structure

Fig. 6. A selection of different 2D planar structures formed in simulation

Figure 5(e) shows that the organism has started disassembling. Clearly, unlike the recruitment process, disassembling can start from more than one point in the organism. After all robots are disconnected from the organism, the "power socket" starts to transmit IR signals again and the cycle is repeated. Each time, the seed robot randomly chooses a pre-defined organism and starts the recruitment procedure. Figure 6 shows some different 2D structures the robots have

constructed within one single simulation run. As the IDs of the robots in the organism are dynamically allocated when they dock, the particular robots that make up the organism vary each cycle. Thus the same robot may play different roles, depending on its position, in different organisms.

6 Conclusions and Future Work

This paper has presented a simple self-organised morphology control mechanism for a group of self-assembling robots. Each robot operates in one of two modes: *Swarm Mode* or *Organism Mode*, and acts accordingly following the rules of a common behaviour-based controller. The autonomous morphogenesis approach is completely decentralised and self-organised with local IR-based robot-robot communications. The 2D planar organism structures are represented with ID-based tree structures. Two node lists (arrays), *sortedNodeList* and *recruitmentNodeList* are generated by a pre-order walk through the corresponding tree representation. Together with the local communication protocols these two node lists, stored in each robot, give sufficient information for the swarm to self-assemble into a specific 2D organism. Each robot is dynamically allocated an ID when it has docked with the developing organism

The proposed morphology control mechanism has been demonstrated using the simulation tool Stage. A simulated robot has been modelled with the same sensing and communication capabilities for docking and recruitment as those of the real SYMBRION robot. Simulation shows that these robots can successfully self-assemble into the specified organism structure. Given the hardware constraints, in a 2D environment, the shapes can be any of those defined in tree structures with fewer than 3 children and no cycles. When very simple disassembly strategies are applied, re-shaping between different organisms can also be achieved using the same controller framework. To improve the energy efficiency of the re-shaping procedure, more complex disassembly strategies need to be investigated in future work. Moreover, as only one robot at a time is allowed to dock during the recruitment process, the efficiency of the algorithm could be further improved by allowing parallel docking. Note also that at the time of writing the algorithm is not fault tolerant and there are many ways in which faults might disrupt the self-assembly process including, for instance, mechanical failure of the docking mechanism or failure of the power or communications busses across the docking mechanism. With real hardware operating over extended periods and multiple robots the probability of such faults is likely to be high. Thus planned work also includes extending the morphogenesis algorithm so that if faults are detected during self-assembly, the process modifies itself to compensate for those faults.

Acknowledgments. The SYMBRION project is funded by the European Commission within the work programme Future and Emergent Technologies Proactive under grant agreement no. 216342.

References

1. Christensen, A., O'Grady, R., Dorigo, M.: Swarmorph-script: a language for arbitrary morphology generation in self-assembling robots. Swarm Intelligence 2(2), 143–165 (2008)
2. Grushin, A., Reggia, J.A.: Automated design of distributed control rules for the self-assembly of pre-specified artificial structures. Robotics and Autonomous Systems 56(4), 334–359 (2008)
3. Guo, H., Meng, Y., Jin, Y.: A cellular mechanism for multi-robot construction via evolutionary multi-objective optimization of a gene regulatory network. Biosystems 98(3), 193–203 (2009)
4. Kernbach, S., Meister, E., Scholz, O., Humza, R., Liedke, J., Ricotti, L., Jemai, J., Havlik, J., Liu, W.: Evolutionary robotics: The next-generation-platform for on-line and on-board artificial evolution. In: Proc. IEEE Congress on Evolutionary Computation, Trondheim, Norway, pp. 1079–1086 (May 2009)
5. Levi, P., Kernbach, S. (eds.): Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution. Springer, Heidelberg (2010)
6. Liu, W., Winfield, A.: Implementation of an IR approach for autonomous docking in a self-configurable robotics system. In: Proc. Towards Autonomous Robotic Systems, Londonderry, UK, pp. 251–258 (September 2009)
7. Murata, S., Kakomura, K., Kurokawa, H.: Toward a scalable modular robotic system. IEEE Robotics Automation Magazine 14(4), 56–63 (2007)
8. Rubenstein, M., Payne, K., Will, P., Shen, W.M.: Docking among independent and autonomous CONRO self-reconfigurable robots. In: Proc. IEEE International Conference on Robotics and Automation, vol. 3, pp. 2877–2882 (May 2004)
9. Salemi, B., Moll, M., Shen, W.M.: SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system. In: Proc. Int. Conf. on Intelligent Robots and Systems, Beijing, China, pp. 3636–3641 (October 2006)
10. Shen, W.M., Salemi, B., Will, P.: Hormone-inspired adaptive communication and distributed control for CONRO self-reconfigurable robots. IEEE Transactions on Robotics and Automation 18(5), 700–712 (2002)
11. Støy, K.: Using cellular automata and gradients to control self-reconfiguration. Robotics and Autonomous Systems 54, 135–141 (2006)
12. Vaughan, R.: Massively multi-robot simulation in stage. Swarm Intelligence 2(2-4), 189–208 (2008)
13. Yim, M., Zhang, Y., Roufas, K., Duff, D., Eldershaw, C.: Connecting and disconnecting for chain self-reconfiguration with Polybot. IEEE/ASME Transactions on Mechatronics 7(4), 442–451 (2002)

Autonomous Multi-agent Cycle Based Patrolling

Yotam Elor and Alfred M. Bruckstein

Faculty of Computer Science and the Goldstein UAV and Satellite Center, Israel
{yotame,freddy}@cs.technion.ac.il

Abstract. We introduce a novel multi-agent patrolling strategy. By assumption, the swarm of agents performing the task consists of very low capability ant-like agents. The agents have little memory, they can not communicate and their sensing abilities are local. Furthermore, the agents do not possess any knowledge regarding the graph or the swarm size. However, the agents may mark the graph vertices with pheromone stamps which can later be sensed. These markings are used as a primitive form of distributed memory and communication. The proposed strategy is a bundle of two algorithms. A single agent (the "leader") is responsible of finding a short cycle which covers the graph, and this is achieved using a "cycle finding" algorithm. All other agents follow that cycle while maintaining even gaps between them using a "spreading algorithm". We prove that the algorithms converge within a finite expected time. After convergence, the maximum time lag between two successive visits to any vertex using the proposed strategy is at most $4\frac{k}{k-1}\frac{l_{max}}{l_{min}}$ times the optimal, where the optimal time is bounded from below by $\frac{n \cdot l_{min}}{k}$, and where l_{max} (l_{min}) is the longest (shortest) edge in the graph and k is the swarm size.

The "cycle finding" algorithm is robust i.e. in case the graph changes, the leader autonomously finds a new patrolling route. The "spreading algorithm" is scalable and robust. In case the patrolling cycle or the number of agents change during run (e.g. as a result of an agent break down) the agents autonomously redeploy uniformly over the patrolling cycle.

1 Introduction and Related Work

The purpose of patrolling is to visit every point in the area as often as possible. It is convenient to model the area being patrolled as an undirected graph. In this model, time is discrete and will be measured in cycles. Using the undirected graph model we can loosely define the patrolling task as continuously visiting all the vertices of a graph. The *idleness* of a vertex is defined as the time since the last visit to this vertex by an agent. In this work we shall use the *worst idleness* evaluation criterion, as defined in [1,2].

Ant-like agents having limited capabilities are considered. The agents are assumed to have little memory ($O(\log_2 n)$ bits, where n is an upper bound on the graph size), the agents have local sensing abilities and no communication is allowed between them. We also assume that the agents do not possess any global

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 119–130, 2010.

information e.g. the size of the graph, its structure, and the number of agents are unknown. However, the agents can mark nodes with time stamps which can be read by other agents later. The markings used are very simple, including only three time stamps per vertex.

We propose a cycle based patrolling strategy i.e. all agents will follow a single cycle which covers the graph. The patrolling task is divided into two sub-tasks. The first sub-task is to find a good patrolling cycle and the second is to deploy the agents uniformly over it. A uniform spread is desired because the *worst idleness* is minimized when the time gaps between the agents following the cycle are equal. In order to find a good patrolling cycle it is proposed to use a single agent, the "leader", following one of the cycle finding algorithms described in Section 3. The leader's responsibilities are to find a good patrolling cycle, maintain the cycle and mark it so other agents will be able to follow it. All other agents follow the cycle while maintaining even gaps between them.

The advantage of the proposed patrolling strategy over previously suggested cycle based algorithms [5,13,3] is its robustness to agent failures and environment changes. In case the graph changes and the patrolling cycle is broken, the leader autonomously finds a new patrolling cycle. In case the patrolling cycle, or the swarm size changes during run, the agents autonomously rearrange in a new uniform formation. It is important to note that the algorithms presented in [5,13,3] are not distributed. On the other hand, the time required to find a patrolling cycle and to spread the agents over it is longer and the patrol quality might be poorer compared to the non-distributed algorithms.

Both the cycle finding algorithm and the deployment algorithm can be used separately. For example, the cycle finding algorithm can be used as a single agent patrolling strategy or as a heuristic to find Hamilton cycles in graphs, see the work of Wagner *et al.* [15]. Assuming a patrolling cycle was marked on the graph in advance (for example, using a TSP approximation algorithm) or in case of a patrol around a close perimeter [1], the deployment algorithms proposed in this work can be used.

Due to space limitations, the proofs are omitted from this paper, they can be found in our readily available TR [6].

2 Previous Work

Chevaleyre *et al.* argued that the optimal single agent patrolling strategy is to follow the shortest TSP circuit [4]. Elmaliach *et al.* proposed a non-distributed algorithm which finds the shortest Hamilton cycle in weighted grids [5]. This cycle may then be used to patrol the graph. In order to spread the agents uniformly over the cycle, they have proposed a non-distributed algorithm which calculates the fastest way to spread the agents uniformly starting from any given formation. Both algorithms require to know the graph in advance and the spreading algorithm requires to know all agents location. Another approach is to divide the graph between the agents and to let each agent patrol only the sub-graph assigned to it. In a previous work we have proposed such a distributed algorithm inspired by physical gas filled

balloons [8]. When direct communication is allowed a "vertex-market" approach can be employed [2,12].

The scenario where the agents can sense the *idleness* of all the vertices of the graph, led to several *heuristic* approaches [11,2]. Two different assumptions were made: either each agent can sense only the vertices' *idleness* relative to it's own visits or each agent sense the vertices' *idleness* with respect to all agents. Note that in the second case the *idleness* of the vertices serves as an implicit communication channel. A distributed sensing variation is to choose the vertex with the highest *idleness* from the adjacent vertices, inspired by ants foraging in an environment using pheromone gradient [16].

Some recent papers have focused on different evaluation criteria [13,1]. Assuming an intruder model the evaluation criterion used is, roughly speaking, the ability to stop the intruder. The main theme of these patrolling strategies is to use probabilistic algorithms in order to avoid static patrolling patterns which could, in an adversarial scenario, be exploited by intruders. However, stopping intruders is only one aspect of patrolling. For applications such as cleaning, maintenance, surveillance or guarding against weak intruders, the *worst idleness* criterion remains very important.

3 Finding Good Patrolling Routes

In this section we consider the first sub-task i.e. finding a good patrolling cycle. Denote the number of vertices in the graph by n. For brevity, it is first assumed that the graph is unweighted. The weighted case is discussed in Section 3.3. Three single-agent algorithms are presented. The first algorithm (PVAW) is designed to patrol Hamiltonian graphs. By executing the proposed algorithm, the agent finds a Hamilton cycle in the graph. This cycle is the shortest among all cycles hence yielding an optimal patrol. The second algorithm (PVAW2) aims to patrol graphs whose square is Hamiltonian. Using the algorithm, the agent finds a cycle of length at most $2n$. The third algorithm (PVAW3) is designed for general graphs. An agent following PVAW3 will find a cycle of length at most $4n$.

3.1 Hamiltonian Graphs

The proposed algorithms are based on the Vertex-Ant-Walk algorithm [16]. VAW is a simple patrolling procedure in which at every time cycle the agent takes a step to the neighbor vertex with the highest *idleness*. An agent performing VAW will eventually follow a cycle which covers the graph. However, this end-cycle is not necessarily simple thus not optimal. It was shown by Wagner *et al.* [15] that a Hamilton cycle is a possible end-cycle of the VAW process. However, VAW sometimes converges to non-Hamiltonian end-cycles (see Figure 7 in [16]). The Probabilistic-VAW (PVAW) algorithm is proposed as an expansion of VAW. An agent performing PVAW acts most of the time according to VAW. However, using only the time stamps, the agent knows when the cycle it follows is not Hamiltonian so the agent performs random steps in order to find a better end-cycle.

Algorithm 1. PVAW

```
   /* the agent is on vertex u                                          */
1  if PrevDiff ≠ 1 and x ≤ p then
2  |  go to a random neighbor of u
3  else
4  |  go to the neighbor of u with the lowest σ value (brake ties randomly)
   /* the agent is on vertex v                                          */
5  PrevDiff ← σ(v) − σ_mem
6  σ_mem ← σ(v)
7  σ(v) ← t
8  t ← t + 1
```

An informal description of PVAW is the following: During the patrol, using only the time stamps, the agent knows if the current vertex and the previously visited vertex were visited consecutively and in the same order at the previous times they were visited. If its true, the agent performs the regular VAW i.e. goes to the neighboring vertex with the lowest time stamp. Otherwise, with a small probability p the agent goes to a random neighbor, or takes a regular VAW step with probability $1 - p$. When the agent follows a Hamilton cycle, the vertices are always visited in the same order so the agent continues to follow the cycle indefinitely. When the agent is not following a Hamilton cycle, there will be infinitely many events of a small probability to change the agent route. Hence the agent will eventually change its route.

PVAW is presented here as Algorithm 1. $\sigma(\cdot)$ are the time stamps on the vertices; $0 < p < 1$ is a constant parameter; x is a random variable chosen uniformly in $[0, 1]$; σ_{mem} and $PrevDiff$ are variables in the agent's memory. In the algorithm as presented, the time stamps are unbounded. In practice, the time can be calculated modulo N (where N is a bound on the graph size) or alternately, the algorithm can be implemented using pheromone decay.

Due to space limitations the proof of Theorem 1 is omitted. It can be found in our TR [6]. Roughly speaking, Theorem 1 is proved by showing that (1) from any system state there is positive probability that the agent will find a Hamilton cycle within a finite time; and (2) once a Hamiltonian cycle was found, the agent will continue to follow it.

Theorem 1. *An agent performing PVAW on a Hamiltonian graph G will eventually patrol the graph using a Hamilton cycle.*

3.2 Non-Hamiltonian Graphs

In this section PVAW is expanded to address graphs which are not Hamiltonian. We first present the algorithm PVAW2 which is an extension of PVAW designed to patrol graphs whose square is Hamiltonian. We denote by G^2 (the square of G) the graph on the vertices of G in which two vertices are adjacent if and only if they have distance of at most 2 in G. Note that many of the graphs being

patrolled are two-connected[1] and "the square of every two-connected graph is Hamiltonian" [10]. Consider an agent who can sense its surrounding to a distance of two edges (instead of one) and can travel two edges in a single time step (again, instead of one). Such an agent performing PVAW on G, while choosing its next step from a 2-neighborhood[2], is practically performing PVAW on G^2. Since G^2 is Hamiltonian, the resulting end-cycle is Hamiltonian.

In the patrolling problem it is usually assumed that an agent can travel only one edge per time cycle. So we consider a more reasonable agent model in which the agent can sense to a distance of two edges, but can travel only a single edge per time cycle. This agent can execute PVAW2 (presented as Algorithm 3 in [6]). PVAW2 is identical to PVAW but in lines 2 and 4 the next vertex is chosen out of a 2-neighborhood. Note that t in PVAW2 does not comply with real time because moving from vertex to vertex might take one or two time cycles. The proof of Theorem 2 is based on Theorem 1.

Theorem 2. *For any graph G such as G^2 is Hamiltonian, an agent performing PVAW2 on G will eventually patrol the graph using a cycle of length at most $2n$.*

PVAW3 is an extension of PVAW designed to patrol any graph. Consider an agent who can sense its surrounding to a distance of 4 edges. Such an agent can perform PVAW on G^4. Since the square of any graph is two-connected (see [6]) then the square of the square of any graph (G^4) is Hamiltonian. So an agent following PVAW3 will find a cycle of length at most $4n$. The pseudo code of PVAW3 with the convergence proof can be found in our TR [6].

3.3 Weighted Graphs

Assume that the graph being patrolled is weighted i.e. every edge has a weight (or length) which is the amount of time required to traverse it. The algorithms PVAW, PVAW2 and PVAW3 can be used to find patrolling cycles in weighted graphs by simply ignoring edge weights. However, for weighted graphs, the quality of the patrolling cycles found is poorer. Let l_{max} (l_{min}) be the length of the longest (shortest) edge in the graph. Let OPT be the length of the shortest cycle which covers the graph. Clearly, $OPT \geq n \cdot l_{min}$. The length of the cycle found by PVAW, for example, is n edges. Hence its length is at most $n \cdot l_{max}$. So PVAW finds a cycle of length at most $\frac{l_{max}}{l_{min}} OPT$. Similarly, PVAW2 finds a cycle of length at most $2\frac{l_{max}}{l_{min}} OPT$ and PVAW3 - $4\frac{l_{max}}{l_{min}} OPT$. Heuristics can be used in order to obtain shorter cycles.

3.4 Discussion and Simulations

Before performing simulations, the value of p (the random step probability) must be set. p should be small enough to allow VAW to stabilize. On the other hand,

[1] A graph G is two connected iff for every vertex $v \in G$, $G \backslash v$ is connected.
[2] The d-neighborhood of vertex u include all the vertices of distance d or less from u.

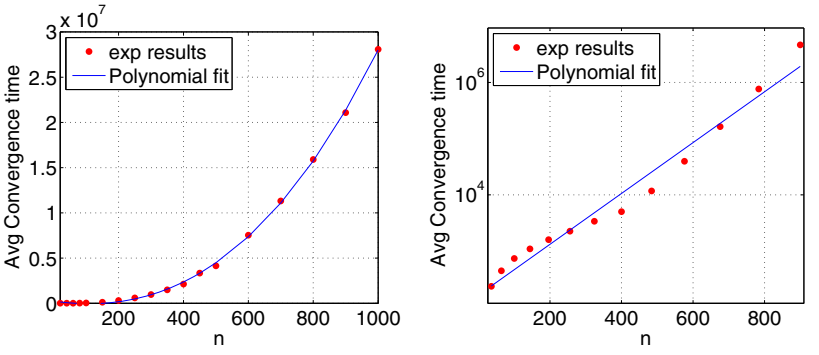


Fig. 1. Convergence time of PVAW3 on random trees (left) and grids (right). Note that the right graph y-axis is logarithmic.

choosing p too small will increase the convergence time because escaping from non-Hamilton end-cycles will take a long time. In our experiments $p = 1/n$ was used.

PVAW3 is the most general of the algorithms described, it can be used to find a patrolling cycle on any graph, so the experiments were focused on it. The experiments were done over unweighted grids and unweighted random trees. A random tree is created by first constructing n vertices and then randomly connecting vertices belong to different components until a single component remains. We have performed experiments with graph size varying from 20 to 1000 vertices. The convergence time of every experiment setup was averaged over 500 runs.

The average time to find a patrolling cycle in a random tree was found to be about $0.04 \cdot n^3$, see Figure 1. The convergence time for grids was found to be exponential, about $e^{n/100}$, see Figure 1. Because grids are denser than trees, they contain more Hamilton cycles and more ways to improve the agent route so we have expected (wrongly) that convergence will be faster on grids. Which graph characteristics effect the convergence rate is currently unknown. We leave that as future work.

4 Swarm Deployment

In this section we consider the second sub-task which is spreading the agents uniformly over the previously found cycle. Algorithm 2 which aims to spread the agents uniformly over a ring is presented. A swarm of agents following Algorithm 2 on a ring will reach a uniform formation within a finite expected time. After reaching the uniform formation, the agents will patrol the ring as efficiently as possible.

A group of k agents on an oriented weighted ring is considered. Every edge e has an integer weight w_e which represents the number of time cycles required to traverse the edge. Let m be the total length of the ring, i.e. $m \triangleq \sum w_e$.

The notations are adopted from [9]. Without losing generality, assume that the agents patrol the ring clockwise. Whenever we fix an agent a, let a_1 (a_{-1}) be the agent who is the clockwise (counter-clockwise) neighbor of a. Similarly a_2 is the clockwise neighbor of a_1 and so on. Let $d(a,b)$ be the distance between agents a and b defined as the sum edge weights on the clockwise path from a to b. In case agents a or b are in the middle of an edge, only the relevant part of the edge is taken into account. Note that $d(a,b) + d(b,a) = m$. The following definitions are very convenient. For agent a,

$$d_{-1}(a) \triangleq d(a_{-1}, a), \quad d_1(a) \triangleq d(a, a_1), \quad d_2(a) \triangleq d(a_1, a_2) \tag{1}$$

A *synchronous* model which is very close to the model of Suzuki *et al.* [14] is considered. In the *synchronous* model the agents are operating in rounds. In every round, every agent sense the environment and takes an action. As opposed to the model by Suzuki *et al.* it is assumed that in every round the agents have slightly different timing phase so they never work simultaneously. In particular, upon sensing neighboring vertices, every agent can sense all time stamps that were made earlier.

In scenarios where agents can sense the location of other agents directly, it is realistic to assume that the agents' sensing range is limited. We have shown in a related work [9] that if the agents' sensing range is shorter than $\lfloor n/k \rfloor$, the agents can not spread uniformly over the ring. The distance estimation mechanism described here is intrinsically not range limited so it enables the agents to spread over rings of any size.

We have considered a very close task in [7]. As in the current scenario, in [7] the agents are required to minimize the *worst idleness* of a ring graph. In [7], it is assumed that every agent can measure the distance to its clockwise and counter-clockwise neighbors (d_1 and d_{-1}). In the current scenario the agents' sensing abilities are poorer, every agent can estimate the distance to the two agents preceding him (d_1 and d_2). To be clear, the sensing model is different in two ways. First, instead of addressing d_{-1} and d_1, every agent addresses d_1 and d_2. Second, instead of measuring distances, the agents can only estimate them using time stamps. The distance estimation mechanism will be described later.

An agent can move clockwise or counter-clockwise on the cycle hence changing the agents formation. For example, if agent a takes a clockwise step while all other agents hold position, $d_{-1}(a)$ is increased and $d_1(a)$ is decreased by one edge. All other inter-agent distances remain unchanged. So every agent is limited to moving vertices between d_{-1} and d_1. In the deployment algorithm in [7], every agent tries to balance d_{-1} and d_1 by moving vertices between them. In the current scenario d_{-1} is unknown to the agents so instead every agent tries to balance d_1 and d_2. Unfortunately, the agents are limited to manipulate only d_{-1} and d_1. This limitation hardens the task because the agents can not effect d_2 and, even worse, a side effect of changing d_1 is changing d_{-1}. Agents are forced to change d_{-1} without being able to sense it. For example, an agent can move vertices from d_{-1} to d_1 even though d_{-1} is smaller than d_1 hence making the deployment less balanced. That is the main reason the task considered here is harder and the algorithms are less efficient when compared to ones discussed in [7] .

When an agent is occupying a vertex, it estimates the distances d_1 and d_2 in the following manner. Denote agent a's estimation of $d_1(a)$ by $\delta_1(a)$, and of $d_2(a)$ by $\delta_2(a)$. In our algorithms, every vertex is marked by two time stamps τ_1 and τ_2 which are the times of the last visit and the second to last visit to the vertex respectively. These markings are maintained by the agents. Consider agent a occupying the vertex u at time t, the distances are estimated by:

$$\delta_1(a) \triangleq t - \tau_1(u) \qquad \delta_2(a) \triangleq \tau_1(u) - \tau_2(u) \tag{2}$$

where t is current time. As shown in [6], δ_1 is an upper bound on d_1 and $\delta_1 + \delta_2$ is an upper bound on $d_1 + d_2$. Unfortunately, δ_2 might be smaller or larger than d_2. Note that the agents are estimating distances only when occupying vertices so $\delta_1(a)$ and $\delta_2(a)$ are undefined when agent a is not on a vertex.

For the analysis purpose, δ_1 and δ_2 are defined for all vertices (and not only for vertices which are occupied by agents). δ_2 definition, given in Equation 2, holds for unoccupied vertices. Consider vertex u which may be occupied or unoccupied at time t. Let $d(a,u)$ be the length of the clockwise path from agent a to vertex u and let $\Delta(u) = t + \min_a \{d(a,u)\}$ where t is current time. Intuitively, $\Delta(u)$ is the next time vertex u will be visited by an agent assuming the next agent to visit vertex u will advance without stopping. Let $\delta_1(u) = \Delta(u) - \tau_1(u)$. It is important to note that in case there is an agent on vertex u then $\Delta(u) = t$ and the definition given in Equation 2 holds. We emphasize again that the quantities $\delta_1(u)$ and $\delta_2(u)$ are defined for all vertices for the analysis purpose. In practice, $\delta_1(u)$ and $\delta_2(u)$ are known only to the agent currently occupying vertex u.

Let $\phi = \max_{u \in G, i \in \{1,2\}} \{\delta_i(u)\}$. The idleness of any vertex u is bounded from above by $\delta_1(u)$. Hence the *worst idleness* is bounded by ϕ.

4.1 The Deployment Algorithm

The following spreading algorithm is proposed (compare with Algorithm 3 in [7]). Every agent follows the ring clockwise while trying to balance δ_1 and δ_2. Recall that the agent can not change δ_2. However, it can enlarge δ_1 by staying a time cycle in its current location. Formally, if $\delta_1 \geq \delta_2$ the agent continues to move clockwise. In case $\delta_1 < \delta_2$ there is a probability $(f(\delta_1, \delta_2))$ that the agent will stop for a time cycle hence enlarging δ_1 (and possibly d_1) by one edge. A pseudo code of the algorithm is presented as Algorithm 2. We prove that for every function $f(\delta_1, \delta_2)$ fulfilling the conditions in Definition 1 the swarm will eventually patrol the ring efficiently i.e. the *worst idleness* will be $\lceil m/k \rceil$.

Definition 1. *The conditions on $f(\delta_1, \delta_2)$ that guarantee convergence are:*

1. *For $\delta_1 < \delta_2$, $0 < f(\delta_1, \delta_2)$.*
2. *For $\delta_1 = \delta_2 - 1$, $f(\delta_1, \delta_2) < 1$.*

Note that for $\delta_1 = \delta_2 - 1$, $0 < f(\delta_1, \delta_2) < 1$ hence the randomness is a necessity.

In the algorithm, as presented in Algorithm 2, the agents share a common notion of time. However, the agents only address the time difference between two consecutive visits to the same vertex. Hence instead of a common time notion

Algorithm 2. Deployment (weighted ring), x is a random variable taken uniformly from $[0,1]$, t is current time

$\delta_1 \leftarrow t - \tau_1$; $\delta_2 \leftarrow \tau_1 - \tau_2$

$p \leftarrow \begin{cases} f(\delta_1, \delta_2) \; if \delta_1 < \delta_2 \\ 0 \; else \end{cases}$

if $x > p$ **then**

  $\tau_2 \leftarrow \tau_1$; $\tau_1 \leftarrow t$

  Go to the next vertex on the ring (might take more than one time cycle).

the agents can use unsynchronized clocks at the vertices or measure pheromone decay.

Due to space limitations, the proof of Theorem 3 is omitted. It can be found in [6]. The Theorem is proved by showing that (1) ϕ is non-increasing; and (2) from any system state such as $\phi > \lceil \frac{m}{k} \rceil$, there is a positive probability that ϕ will decrease within a finite time.

Theorem 3. *Consider a swarm of k agents following Algorithm 2 on a weighted ring of size m. The agents will reach a formation in which* $\phi = \lceil m/k \rceil$ *within a finite expected time.*

4.2 Simulations

The simulations were conducted over unweighted rings of varying size and with varying number of agents. $f(\delta_1, \delta_2)$ was of the form

$$f(\delta_1, \delta_2) = \begin{cases} 1 \text{ if } \delta_1 < \delta_2 - 1 \\ c \text{ if } \delta_1 = \delta_2 - 1 \\ 0 \text{ else} \end{cases} \tag{3}$$

with $c = 1/n$. Before starting the algorithm, all agents were placed on a single vertex and for all vertices τ_1 and τ_2 were set to zero. So initially $\phi = n$. Let $T(x)$ be the time in which the swarm achieve $\phi = x$. All results were averaged over 500 runs.

$T\left(\lceil \frac{n}{k} \rceil + \frac{k}{2}\right), T\left(\lceil \frac{n}{k} \rceil + 1\right)$ and $T\left(\lceil \frac{n}{k} \rceil\right)$ for a swarm of 10 agents spreading over rings of size $98-112$ vertices are displayed in Figure 2. Observe that $T\left(\lceil \frac{n}{k} \rceil\right)$ is very sensitive to the value of $(n) \bmod k$. In case $(n) \bmod k = 0$, $T\left(\lceil \frac{n}{k} \rceil\right)$ is very high compared to any other value of $(n) \bmod k$. On the contrary, $T\left(\lceil \frac{n}{k} \rceil + 1\right)$ is not sensitive to $(n) \bmod k$ and is much smaller than $T\left(\lceil \frac{n}{k} \rceil\right)$. A typical run of the algorithm with 10 agents on a ring of size 100 (i.e. $(n) \bmod k = 0$) is presented in Figure 2. In a typical run, ϕ decreases very rapidly until reaching $\phi = \lceil \frac{n}{k} \rceil + \frac{k}{2}$. $\phi = \lceil \frac{n}{k} \rceil + 1$ is achieved shortly after and $\phi = \lceil \frac{n}{k} \rceil$ is achieved much later. Because the patrolling quality is almost optimal when $\phi = \lceil \frac{n}{k} \rceil + 1$ and $T\left(\lceil \frac{n}{k} \rceil + 1\right)$ is not sensitive to small changes in n, we focus our experiments on $T\left(\lceil \frac{n}{k} \rceil + 1\right)$.

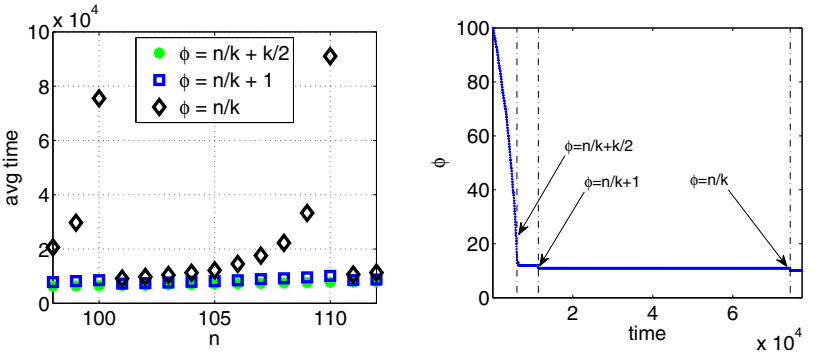


Fig. 2. (left) $T\left(\left\lceil\frac{n}{k}\right\rceil+\frac{k}{2}\right), T\left(\left\lceil\frac{n}{k}\right\rceil+1\right)$ and $T\left(\left\lceil\frac{n}{k}\right\rceil\right)$ for a swarm of 10 agents spreading over rings of size $98-112$ vertices. (right) A typical run of 10 agents on a 100 vertices ring.

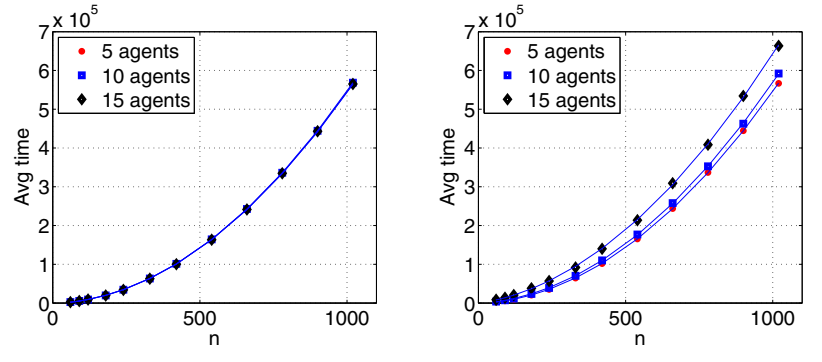


Fig. 3. (left) Average time to achieve $\phi=\left\lceil\frac{n}{k}\right\rceil+\frac{k}{2}$. (right) Average time to achieve $\phi=\left\lceil\frac{n}{k}\right\rceil+1$.

$T\left(\left\lceil\frac{n}{k}\right\rceil+\frac{k}{2}\right)$ and $T\left(\left\lceil\frac{n}{k}\right\rceil+1\right)$ are presented in Figure 3. $T\left(\left\lceil\frac{n}{k}\right\rceil+\frac{k}{2}\right)$ is not dependent on the number of agents and is found to be about $\frac{1}{2}\cdot n^2$. $T\left(\left\lceil\frac{n}{k}\right\rceil+1\right)$ is effected by the number of agents. The larger the swarm, the larger $T\left(\left\lceil\frac{n}{k}\right\rceil+1\right)$. Nevertheless, for any swarm size $T\left(\left\lceil\frac{n}{k}\right\rceil+1\right)$ was found to be $\Theta\left(n^2\right)$.

5 Bundling the Algorithms

In the two previous sections we have shown how to find a good patrolling cycle and how to deploy agents over a ring. In this section we bundle the two algorithms to create the multi agent patrolling strategy. When following the proposed strategy, there is one agent acting as leader. All other agents (denoted as the "herd") follow the cycle marked by the leader. All agents start the algorithm at the same time, meaning that the agents of the herd will try to spread uniformly

before the leader have found a cycle. Clearly, they will fail. Nevertheless, after a stable cycle was reached, the herd will spread uniformly over it.

It is assumed that all agents have the same sensing abilities as the leader. For example, if the leader execute PVAW2 then all agents can sense to a distance of two edges. For this case, it was shown in [6] that the agents can follow the patrolling cycle marked by the leader. Thus we simplify the problem from spreading over a non-simple cycle to spreading over an oriented weighted ring.

For generality, we assume the leader executes PVAW3. Considering Hamiltonian or 2-connected graphs, similar results can be obtained for PVAW or PVAW2 respectively. When applying the proposed patrolling strategy, the leader will find a patrolling cycle of length at most $4n \cdot l_{max}$ within a finite expected time. After the leader has found a stable cycle, the herd, following Algorithm 2, will spread uniformly over it. After stabilization, the *worst idleness* obtained by the $k-1$ agents of the herd is at most $\lceil 4n \cdot l_{max} / (k-1) \rceil$. Hence the *worst idleness* of our proposed strategy is at most $4\frac{l_{max}}{l_{min}} \frac{k}{k-1} \cdot OPT_{wi}$ where OPT_{wi} is the *worst idleness* of the optimal strategy, where we have used the trivial bound of $OPT_{wi} \geq \frac{n \cdot l_{min}}{k}$.

In case some of the agents of the herd break down, the remaining active agents autonomously rearrange in a new uniform formation. In case the leader breaks down, one of the herd agents should be called upon to replace it. The agents of the herd can notice that there is no leader around because the σ stamps get old. However, a special attention is required in order to avoid situations with two or more leaders in swarm. For example, a distributed leader election algorithm can be used. We leave the issue of robustness to leader breakdowns as a future work.

6 Conclusion

In this paper we have presented a patrolling strategy composed of a bundle of two algorithms. The leader agent is responsible of finding and marking a good patrolling cycle. All other agents (the herd) follow that cycle while maintaining even gaps between them. The *worst idleness* achieved by a swarm of k agents following our strategy is at most $4\frac{k}{k-1}\frac{l_{max}}{l_{min}}$ times the optimal.

In practice, the leader might require exponential time to find a good patrolling cycle. So PVAW algorithms are useful in case we know convergence will be fast (if for example, the graph being patrolled is tree-like) or in case fast convergence is not essential. On the contrary, the deployment algorithm is efficient. An almost-optimal spread was achieved within $O\left(n^2\right)$ time cycles in all scenarios. The deployment algorithm can be used with other "cycle building" algorithm e.g. some form of DFS or a non-distributed TSP approximation algorithm.

A novel pheromone based distance estimation mechanism was presented which is intrinsically not range limited.

Acknowledgments. This research was supported by the Technion Goldstein UAV and Satellite Center and by the European Community's FP7-FET program, SMALL project.

References

1. Agmon, N., Kraus, S., Kaminka, G.A.: Multi-robot perimeter patrol in adversarial settings. In: IEEE Int. Conf. on Robotics and Automation, pp. 2339–2345 (2008)
2. Almeida, A., Ramalho, G., Santana, H., Tedesco, P.A., Menezes, T., Corruble, V., Chevaleyre, Y.: Recent advances on multi-agent patrolling. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 126–138. Springer, Heidelberg (2004)
3. Basilico, N., Gatti, N., Amigoni, F.: Developing a deterministic patrolling strategy for security agents. In: IEEE/WIC/ACM Int. Joint Conf. on Web Intel. and Intel. Agent Technologies, vol. 2, pp. 565–572 (2009)
4. Chevaleyre, Y., Sempe, F., Ramalho, G.: A theoretical analysis of multi-agent patrolling strategies. In: AAMAS, pp. 1524–1525. IEEE Computer Society, Washington (2004)
5. Elmaliach, Y., Agmon, N., Kaminka, G.A.: Multi-robot area patrol under frequency constraints. In: IEEE Int. Conf. on Robotics and Automation, pp. 385–390 (2007)
6. Elor, Y., Bruckstein, A.M.: Autonomous multi-agent cycle based patrolling. Tech. rep., Computer Science Department, Technion Haifa, Israel (September 2009)
7. Elor, Y., Bruckstein, A.M.: Multi-agent deployment and patrolling on a ring graph. Tech. rep., Computer Science Department, Technion Haifa, Israel (September 2009)
8. Elor, Y., Bruckstein, A.M.: Multi-agent graph patrolling and partitioning. In: IEEE/WIC/ACM Int. Joint Conf. on Web Intel. and Intel. Agent Technologies, pp. 52–57 (2009)
9. Elor, Y., Bruckstein, A.M.: Multi-agent deployment on a ring graph. In: Ants 2010: Seventh International Conference on Swarm Intelligence (to appear, 2010)
10. Fleischner, H.: The square of every two-connected graph is hamiltonian. Journal of Combinatorial Theory 16(1) (1974)
11. Machado, A., Ramalho, G., Zucker, J.D., Drogoul, A.: Multi-agent patrolling: An empirical analysis of alternative architectures. In: Sichman, J.S., Bousquet, F., Davidsson, P. (eds.) MABS 2002. LNCS (LNAI), vol. 2581, pp. 155–170. Springer, Heidelberg (2003)
12. Menezes, T., Tedesco, P., Ramalho, G.: Negotiator agents for the patrolling task. In: Sichman, J.S., Coelho, H., Rezende, S.O. (eds.) IBERAMIA 2006 and SBIA 2006. LNCS (LNAI), vol. 4140, pp. 48–57. Springer, Heidelberg (2006)
13. Sak, T., Wainer, J., Goldenstein, S.K.: Probabilistic multiagent patrolling. In: Zaverucha, G., da Costa, A.L. (eds.) SBIA 2008. LNCS (LNAI), vol. 5249, pp. 124–133. Springer, Heidelberg (2008)
14. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM Journal on Computing 28(4), 1347–1363 (1999)
15. Wagner, I.A., Bruckstein, A.M.: Hamiltonian(t) - An Ant-Inspired Heuristic for Recognizing Hamiltonian Graphs. In: Proceedings of the 1999 Congress on Evolutionary Computation (1999)
16. Wagner, I.A., Lindenbaum, M., Bruckstein, A.M.: Efficiently searching a graph by a smell-oriented vertex process. An. of Math and Art. Intel. 24(1-4), 211–223 (1998)

Biologically Realistic Primitives for Engineered Morphogenesis

Justin Werfel

Wyss Institute for Biologically Inspired Engineering
Harvard University, Cambridge, USA
justin.werfel@wyss.harvard.edu

Abstract. Finding ways to engineer morphogenesis in biological systems, to direct the development of a multicellular organism according to desired specifications, will require both high-level understanding of organizing principles in such systems and low-level understanding of how basic tools can be reliably implemented in real cells. Past work has assumed low-level capabilities appropriate to computing agents but not necessarily to biology. Here I discuss potential ways of implementing low-level primitives based on capabilities for which evidence exists in biological systems, with the goal of developing a basis for engineering developmental processes that will be realizable in wetware. I focus on the use of biologically realistic morphogen gradients to produce structures of desired size, provide positional information, and trigger genetic cascades that lead to the growth of more complex structures.

1 Introduction

Morphogenesis, the process of development from a single cell to a complex multicellular organism, is one of the great examples of robust collective behavior. The growth and differentiation of genetically identical cells, reliably resulting in a given high-level structure, is the phenomenon that drives the field of developmental biology. Moving from science toward engineering, the field of synthetic biology seeks to find ways to program cells to make them exhibit desired behaviors. While synthetic biology has so far focused on ensembles of unicellular organisms, its ultimate goal will encompass multicellular organisms, enabling us to grow plants and animals of desired morphology by specifying a genetic program.

Achieving this goal will require programming principles for agents with the unique characteristics of biological cells. Existing studies on the engineering of bioinspired structure-forming systems [1, 2, 8, 13, 16, 18] draw important ideas from principles known to operate in biological systems, but often make assumptions about the systems they consider that may be unrealistic for living cells or difficult to achieve.

This paper uses an agent-based cell model to consider certain primitives that are known to operate in natural developmental systems and likely to be critical to engineered ones, and seeks to develop principles for their implementation

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 131–142, 2010.

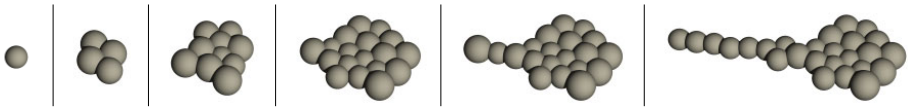


Fig. 1. Snapshots during the growth of a rounded structure with a narrow projection, grown using the model and approach for engineered morphogenesis described here

that may realistically be achievable by synthetic biologists in the near future. These primitives include tissue growth limited to desired dimensions, the establishment of coordinate systems, and the demarcation of well-defined domains of the embryo. In this work I focus on morphogenesis in two-dimensional sheets of cells (Fig. 1), as in epithelial tissues, though the model is intrinsically three-dimensional.

Section 2 discusses related work. Section 3 lays out the assumptions made about cell capabilities. Section 4 describes the agent-based model used to evaluate the approach. Section 5 discusses the primitives addressed here. Section 6 concludes.

2 Related Work

Studies of morphogenesis in natural systems have identified key principles that underlie organismal development [9, 11, 17]. Substances called morphogens direct cell growth and differentiation, triggering different behaviors depending on their concentration. Some morphogens are produced by cells in genetic cascades as the developmental process unfolds, while others are deposited in the egg by the mother before fertilization. Gradients of morphogen concentration, spatially distributed across an embryo or subregion, are the basic tools used to establish locations and directions. Small concentration differences can be amplified to produce stripes and other sharply demarcated expression regions.

Past work specifically concerned with engineering developmental processes [1, 2, 13] has relied on several of these principles. Nagpal [13] describes programming methods for a flat sheet of cells to coordinate folding into any user-specified shape, using axioms from origami. Cells are randomly distributed on the sheet and do not change arrangement or number. Doursat [1, 2] demonstrates methods by which a small initial set of cells can grow into a complicated planar structure with controllable shape and differentiation pattern. These studies have abstracted cells into computational units capable of simple calculations, with morphogen gradients implemented using integer counters, and broadcast messages used to signal completion of developmental stages. While such capabilities are eminently suitable for engineered computing systems, they may be problematic for biological cells, and attempts to implement such systems in synthetic biology will have to rely on capabilities available to real cells.

Other studies that have addressed problems of structure formation in engineered distributed systems generally make additional domain-specific assumptions that

make application to biological cells difficult. Modular robots [18] and collective construction [16] generally require very precise alignment of structural elements, often on a discrete lattice. Programmed self-assembly [8] and collective construction [16] typically use rigid structural units that cannot be rearranged within an assembly once in place. By contrast, cells are variable in size and deformable; they may come together in arbitrary ways, rearrange their configuration at both large and small scales throughout morphogenesis, and produce new units deep within a structure.

An alternative perspective for creating developmental systems with desired characteristics is given by work in evolutionary computation [4, 5]. Such work takes the approach of automated exploration of large search spaces, rather than being concerned with tools and principles for use by human designers.

3 Assumptions

I assume that cells possess the following basic capabilities, based on published studies of biological systems:

1. Measure the local concentration of a morphogen, and respond according to whether the concentration falls above or below some threshold [3, 11]. Responses can be to produce a new morphogen and/or to enable cell growth. Multiple thresholds of response may exist for a single morphogen [3, 7, 17] (e.g., through multiple types of receptors with different affinities), so that the same morphogen may, for instance, evoke one response at high concentrations, a different response at low concentrations, and no response in between.
2. Measure and respond to the local *slope* of a morphogen gradient, i.e., the direction of greatest change and the magnitude of that change [11, 15]. Responses can be to produce a new morphogen, to enable cell growth, and/or to affect the orientation of the cleavage plane during division.
3. Determine whether any other cell borders it in a given direction [6, 12].

I further assume that the sources that result in production of some morphogens can be deposited in the egg by the mother, and remain localized as the embryo grows [11]. Finally, I assume that production, diffusion and degradation of morphogens happens quickly compared to growth and movement of cells. This last assumption can be relaxed to some extent, though I do not explore doing so in detail in this work.

4 Model

Here I describe the agent-based simulation model used to test the approaches to be described in §5. Cells are modeled as three-dimensional entities normally confined to a two-dimensional sheet, as with epithelial tissue. Future work will consider deformations of the sheet.

The model keeps track of each cell's position (real-valued 3D coordinates) and volume, and the amounts of the various morphogens within that volume.

Cells act like deformable, adhesive spheres on an adhesive substrate. Each cell experiences a spring-like force (repulsive at short distances, attractive at longer ones) from each of its neighbors and the substrate, governing its movement in space.

Neighbors can be determined by Voronoi calculation or other methods. For computational efficiency, this model determines which cells border on which others at each time step using a discretized volumetric representation (details omitted for space reasons).

Once neighbors have been identified, forces can be calculated. Pairs of neighboring cells exert opposing forces of magnitude $k(r - r_0)$ on each other along the line connecting their centers, where $k = 1$ is a spring constant, r is the distance between the two cell centers, and r_0 is the "rest distance" based on the sizes of the two cells (if the cells have volumes V_1 and V_2, $r_0 = \sqrt[3]{3V_1/4\pi} + \sqrt[3]{3V_2/4\pi}$). Similarly, the substrate exerts a force on cells that neighbor it, acting as a virtual cell with location directly below the other's center. Each cell's position at each time step is updated by an amount proportional to the net force on it, with proportionality constant 1, which gives qualitatively realistic movement without simulating the complete physics of the system. Cell vertical positions are limited to $[0, 0.2]$ to forbid passing through or losing contact with the substrate.

The level m_{ic} of morphogen i in cell c changes according to synthesis, diffusion/transport [10], and degradation:

$$\frac{\Delta m_{ic}}{\Delta t} = \sum_j s_{ij} f_{ij}(m_{jc}) + D_i \sum_{d \in N_c} (\rho_{id} - \rho_{ic}) - \sum_{j \neq i} d_{ij} m_{ic} m_{jc} - d_i m_{ic} \qquad (1)$$

where s_{ij} is the rate at which morphogen j leads to the synthesis of morphogen i; f is a function that may be, e.g., linear to reflect a constant rate of synthesis, or sigmoidal to reflect cooperative binding (unless otherwise specified, $f(x) = x$); D_i is the rate of transfer of morphogen between neighboring cells with different concentrations; N_c is the set of neighbors of cell c; ρ_{ic} is the concentration of morphogen i in cell c (if V_c is the volume of cell c, $\rho_{ic} = m_{ic}/V_c$); d_{ij} is the rate at which morphogen j leads to the degradation of morphogen i; and d_i is the intrinsic degradation rate of morphogen i.

The gradient of a morphogen at a cell's location is estimated based on the concentration of the morphogen there and at each of the cell's neighbors: $\nabla_{ic} = \sum_{d \in N_c} \frac{\rho_{id} - \rho_{ic}}{|r_{cd}|} \hat{r}_{cd}$, where r_{cd} is the vector from cell c to cell d.

The presence of a morphogen may determine cell growth, as described further in §5. If a cell undergoes growth in a given time step, its volume increases by $G = 0.05$ cubic units. If the volume reaches twice its initial value, the cell commits to division: no further growth occurs for $\tau = 30$ time steps, at the end of which division occurs. A new cell is instantaneously created; both daughter cells have half the volume and half the morphogen levels (therefore the same morphogen concentrations) as the mother, and locations equal to that of the mother plus small opposite offsets along a direction which may be affected by one or more morphogen gradients, as described further in §5.

Simulations are initialized with one cell of volume 1.5 cubic units, with maternally deposited morphogen sources already present as specified in §5. Maternal-effect morphogen sources stay closely localized within an embryo [11], and here are taken to remain entirely with one of the two daughters (the one closer to the edge of the embryo) when a cell divides.

Simulations are written in C and visualized using POV-Ray.

5 Primitives

In this section I discuss how tissue growth to limited dimensions can be achieved, using morphogen gradients along one (§5.1) or two (§5.2) axes. I also outline how these gradients can be used to provide position references for cells in an embryo (§5.3), and how genetic cascades can trigger downstream events to produce more complex structures (§5.4).

5.1 Constrained Growth Using Morphogen Gradients

One critical ability in a developmental system is for a structure to grow out to a given size and then stop growing. Doursat [1, 2] accomplishes this with explicit integer hop counts and broadcast signals. Here I explore how it might be achieved with more realistic consideration of morphogen gradients.

I consider two distinct approaches that involve different spatial concentration profiles, one roughly exponential, the other linear. The primary advantage of the first is that it requires only one morphogen. The primary advantage of the second is that growth to a desired size may be accomplished in logarithmic time rather than linear.

Exponential gradient. A morphogen with a single source, spreading by diffusion and degrading everywhere, will generate an exponentially decaying concentration profile. The length scale is given by the square root of the ratio of the diffusion constant to the decay constant [9]—here, $\sqrt{D_i/d_i}$.

First consider growth starting from a single cell (growth of structures later in morphogenesis will be considered in §5.4). A maternally deposited morphogen source leads to production of a morphogen gradient. Using *bicoid* in *Drosophila* as a motivating example [11, 14], I consider a morphogen M_1 whose production is confined to the anterior pole of the embryo and whose product diffuses posteriorward (Fig. 2A). Production of M_1 occurs at rate 0.05 in the initial cell, and in the more anterior daughter cell each time a cell producing M_1 divides, with no production of M_1 elsewhere. (Formally, the mother deposits a morphogen source M_{-1} at level $m_{-1} = 1$, which is neither produced nor degraded by the embryo, which remains confined to the anterior pole, and which results in the production of M_1 with rate $s_{1,-1} = 0.05$.) Diffusion of M_1 occurs at rate $D_1 = 0.002$, decay at rate $d_1 = 0.001$ everywhere.[1]

[1] These values, and others throughout this paper, are somewhat arbitrarily chosen. Exact parameter choices will affect the exact dimensions of the developing embryo and time course of development, but the qualitative behavior of the developmental process is not sensitive to careful parameter tuning.

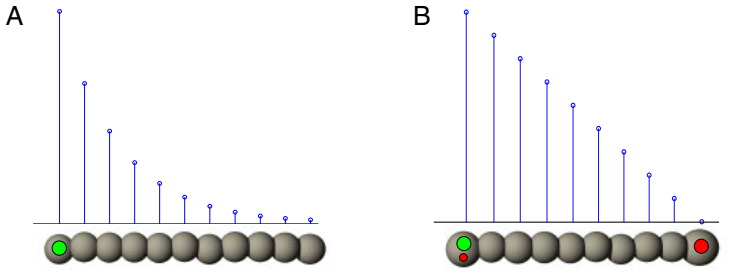


Fig. 2. Maternally deposited morphogen sources can lead to the production of a morphogen gradient, which can be used to regulate growth. A: A source M_{-1} (green) at one end, producing a morphogen M_1 that degrades at constant rate everywhere, results in an exponentially decreasing M_1 concentration profile (stem plot). B: A source M_{-1} (green) at one end producing a diffusible morphogen M_1 that does not degrade on its own, and another source M_{-2} (red) at the other end producing a nondiffusible morphogen M_2 that degrades M_1, results in a linear M_1 gradient. With these two sources alone, the slope is independent of distance between the two (so that the concentration at the left end is proportional to that distance); adding a small M_2 source to the site of the M_1 source limits the M_1 concentration there, resulting in a fixed concentration at the left end and a slope inversely proportional to length.

A cell can grow if both (1) the concentration ρ_1 is high enough and (2) there is no neighboring cell in the "downhill" direction of the concentration gradient. The second condition limits growth to a layer of cells along the posterior edge of the embryo. When division occurs, the cleavage plane is oriented to have its normal vector aligned with the gradient [15].

The final length of the embryo is affected by the concentration threshold at which growth occurs (Fig. 3A) as well as by the diffusion and decay constants D_1 and d_1. The set of possible final lengths will depend on the ability of cells to resolve differences in concentration levels; in particular, the maximum possible final length will depend on the lowest concentration detectable by a cell. Longer structures will require changing the length scale of the gradient via D_1 and d_1, and not merely reducing the threshold for growth.

Because growth is confined to a "leading edge" of cell division, the time required to produce a structure of a given length scales linearly with that length (Fig. 3B). The next section considers a mechanism allowing all cells to divide in parallel, resulting in much faster growth.

Linear gradient. A morphogen produced only at one source and degraded only at one sink will generate a linear concentration profile between the source and sink [9, 17].

A maternally deposited morphogen source M_{-1} confined to the anterior end of the embryo as in the previous section acts as the source for M_1, and a second morphogen M_2 produced by a maternally deposited morphogen source M_{-2}

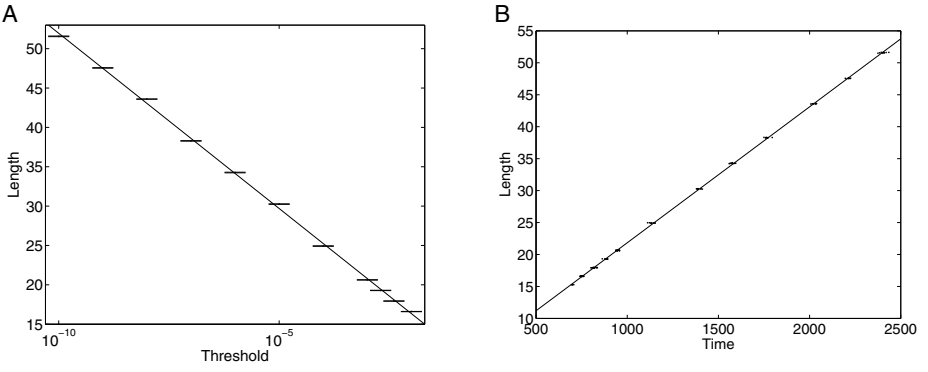


Fig. 3. Length control with an exponential morphogen gradient. A: The length of the embryo can be consistently controlled by setting the morphogen concentration threshold at which growth occurs. The relationship is logarithmic (in this example, dividing the threshold by about 1.7 gives an increase in length of one cell). Averages are over 10 independent trials. B: The time required to grow to a given final length is linearly proportional to that length. Each point represents an independent trial.

confined to the posterior end acts as the sink (Fig. 2B). M_1 is produced at rate $s_{1,-1} = 0.02$ and M_2 at rate $s_{2,-2} = 0.01$. The only effect of M_2 is to degrade M_1, at rate $d_{12} = 0.05$. M_1 does not degrade appreciably on its own ($d_1 = 0$), and M_2 is not transferred to cells where it is not produced ($D_2 = 0$). M_1 diffuses with rate $D_1 = 0.01$; M_2 degrades with rate $d_2 = 0.001$.

With these two maternal deposits M_{-1} and M_{-2}, a linear M_1 gradient of constant slope will develop between the ends of the embryo; the level m_1 at the anterior end will be proportional to the distance between the source and sink. Thus cells at the anterior end can monitor the length of the structure, in a way much like a discrete hop count, and respond in a desired way when the concentration exceeds some threshold. This capability provides a potentially useful tool.

One way to make growth stop when the tissue has reached a desired length, then, is for high levels of m_1 to trigger production of an additional, fast-diffusing morphogen that shuts off growth. This approach requires an extra morphogen, however, and potentially a lag while it diffuses during which growth continues asymmetrically at the end away from the source. A more elegant alternative is to add a second deposit of M_{-2} confined to the anterior end, at half the level of that at the posterior end. The level of M_1 at the anterior end is then dominated by local synthesis and degradation, resulting in a fixed concentration at that site approximately independent of embryo length, and a linear gradient whose slope is inversely proportional to length.

We then specify that a cell can grow if the slope of the M_1 gradient exceeds a given threshold; the choice of threshold, and the values of the synthesis and degradation constants, will determine the final length of the structure. All cells

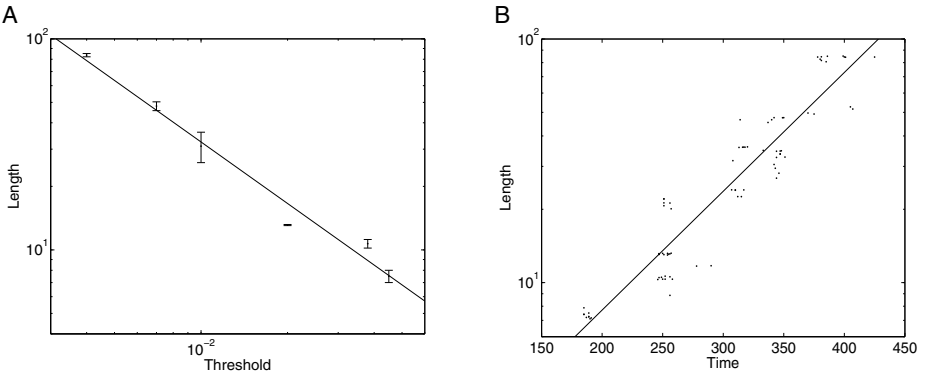


Fig. 4. Length control with a linear morphogen gradient. A: The length of the embryo can be controlled by setting the threshold for slope of the morphogen gradient at which growth occurs. The relationship is geometric (halving the threshold doubles the length of the embryo). Averages are over 10 independent trials. B: The time required to grow to a given final length is proportional to the logarithm of that length. Each point represents an independent trial.

in the embryo can simultaneously contribute to its growth, so that the time required to produce a structure of a given length scales logarithmically with that length (Fig. 4B).

Controlling width via orientation of cell division. When a cell divides, the cell cleavage plane can be influenced by the direction of the gradient [15]. Aligning the plane of division entirely with the gradient leads to growth of the embryo along a straight line; adding a random component to the orientation of the plane results in growth of a wider structure (Fig. 5).

Because the morphogen gradient provides only a local direction reference and not a global one for the developing embryo, stochasticity in the orientation of the cleavage plane can lead to deformation of the overall structure. Fig. 6 shows an

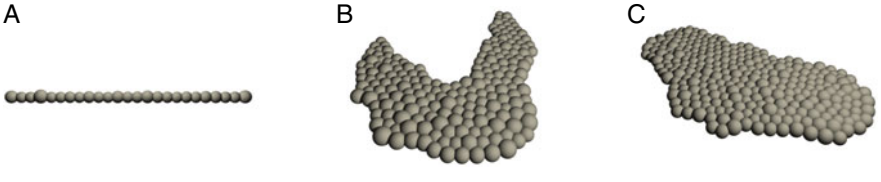


Fig. 5. The extent to which the cleavage plane in cell division aligns with a morphogen gradient affects the width of the resulting structure. A: Division aligned completely with gradient. B: Normal vector to plane of division chosen to be 80% aligned with gradient, 20% randomly oriented. Embryo has grown in a curve as in Fig. 6. C: Division randomly oriented. Examples use a linear morphogen gradient with threshold 0.01.

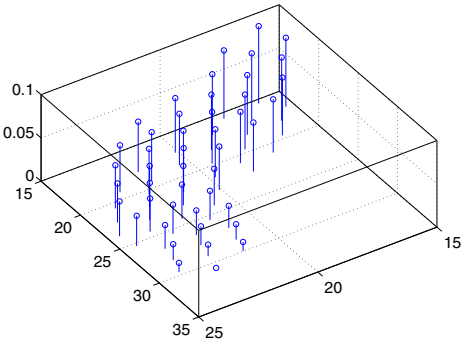


Fig. 6. Embryo grown with a linear morphogen gradient, threshold 0.02, and plane of division chosen to be 80% aligned with gradient, 20% randomly aligned. The plot shows the location of each cell in the x-y plane, and the morphogen concentration ρ_1 on the z-axis.

example where an embryo has grown in an overall curved shape, using a linear gradient as described above. Although the shape of the embryo can change in this way from trial to trial, its intrinsic size remains reasonably consistent: the length of the embryo was found to vary by less than 15% (20 ± 3) in 9 trials, where length was measured along the shortest cell-to-cell path from source to sink (found using A* search).

5.2 Multiple Gradients

Multiple independent morphogen gradients can be present in an embryo at the same time, and direct different aspects of its growth. For instance, anteroposterior and dorsoventral axes can be simultaneously established by separate orthogonal gradients, as is the case in *Drosophila* [11]. Two such gradients can allow growth to desired dimensions in both directions.

Fig. 7A shows an example of an embryo grown in this way, using two linear morphogen gradients. A single cell is initialized with four maternal sources: M_{-1} and M_{-2} forming one axis exactly as in the case described above, and an analogous pair M_{-3} and M_{-4} forming another axis. These sources become confined to individual cells as the embryo grows, as in the single-gradient case discussed above. Each cell evaluates the slope of the two gradients independently, and has separate growth thresholds for each, with growth occurring if either threshold is exceeded. When division occurs, the cleavage plane is chosen along one of the two gradients at random, with greater probability for steeper gradients.

5.3 Position Information

Morphogen gradients provide critical positional information in a developing embryo, allowing cells to undergo differentiation and other key events according to

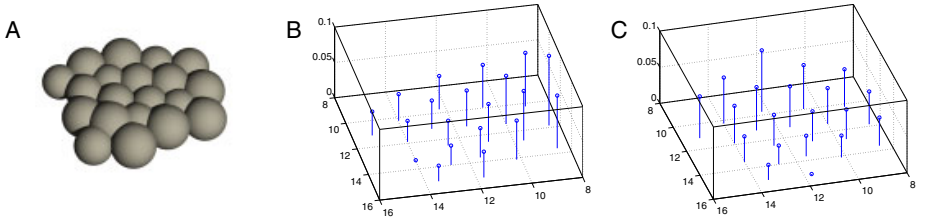


Fig. 7. Growth using two independent morphogen gradients. A: An embryo that developed with separate linear gradients as described in the text. B: Concentration ρ_1 of one morphogen (source at right, sink at left). C: Concentration ρ_3 of the other key morphogen (source at back, sink at front).

their location [11, 17]. Fig. 7B and C shows how two separate morphogen gradients can serve as cues to establish a two-dimensional coordinate system: the concentration of each gradient carries information about a cell's position along the corresponding axis.

Either exponential or linear gradients can be used in principle for this position information. A cell's ability to localize itself will depend on its ability to resolve differences in concentration, and, when in the presence of noise, on the slope of the morphogen gradient. The profile of an exponential gradient, with its relatively steep drop near the source and shallow slope over most of its length, makes it likely to be less useful than a linear gradient in most situations: with a shallower slope, a given error in estimating the concentration will result in a greater position error.

5.4 Cascading Effects

With position information available, cells can differentiate by expressing additional morphogens based on their location in an embryo. Fig. 8 shows an example, where cells in the embryo of Fig. 7 can express five additional morphogens $M_5 \cdots M_9$ depending on the concentrations ρ_1 and ρ_3. Choosing f_{i1} and f_{i3} in Eq. 1 to be sigmoidal functions for $i \in \{5 \cdots 9\}$ allows expression regions to be sharply defined. M_5 is expressed where ρ_3 is high, M_6 where ρ_3 is low; M_7 is expressed for low ρ_1, M_8 for intermediate ρ_1, and M_9 for high ρ_1 (details omitted for space constraints).

These morphogens can have additional downstream effects. For instance, we can build on the program that results in the embryo of Fig. 8, adding a morphogen M_{10} produced where expression levels of M_6 and M_9 are high (purple cells in the figure). Cells for which levels of M_5 and M_7 are high (red in the figure) respond to M_{10} by growing if ρ_{10} exceeds a given threshold and there is no neighboring cell in the downhill direction of the M_{10} gradient. In this way we establish a secondary growth process governed by a morphogen with an exponential concentration profile. The morphogen source is produced by the embryo itself rather than maternally deposited factors, and only a subset of cells respond

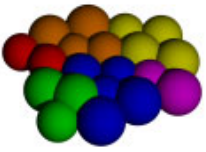


Fig. 8. Expression of downstream morphogens in the embryo of Fig. 7. M_5 expression is high in the red/orange/yellow regions and low elsewhere; M_6 is high in green/blue/purple regions; M_7 is high in red/green regions; M_8 is high in orange/blue regions; M_9 is high in yellow/purple regions.

with growth, leading to a narrow "arm" growing out from the main body of the embryo. The resulting structure is the one shown in Fig. 1.

6 Conclusion

In this paper I have demonstrated approaches to implementing primitives important for engineered morphogenesis, in ways realistically achievable by real biological systems. While actual realization of these primitives in a synthetic biological system remains far from easy, my hope is that the principles discussed here will help point the way toward engineered morphogenetic systems becoming a reality.

Future work will investigate the control of timing issues in genetic cascades, to explore ways of ensuring that events reliably occur in sequence without using explicit global timing signals; and in a likely related effort, exploring ways of incorporating feedback cycles into genetic cascades rather than using strictly feed-forward networks. I also intend to better characterize the mapping between the choice of parameter values and the dimensions of the resulting structure, to allow prediction of structure characteristics from parameters without relying on simulation, and conversely to guide choices of parameter values in order to produce particular desired structures. This last feature will be critical for the eventual goal of global-to-local compilation, the ability to start from a high-level description of a specific user-defined organism and automatically generate a genetic program guaranteed to make a cell develop into that organism. Global-to-local compilation has been achieved in certain related domains [13, 16] but remains a challenge for general morphogenetic systems. Another near-term direction for future work is to extend the model to allow three-dimensional deformation of an epithelial sheet, to begin to incorporate gastrulation and other three-dimensional processes in real morphogenesis into the repertoire of tools available to synthetic systems.

References

1. Doursat, R.: Organically grown architectures: Creating decentralized, autonomous systems by embryomorphic engineering. In: Würtz, R.P. (ed.) Organic Computing, pp. 167–200. Springer, Heidelberg (2008)
2. Doursat, R.: Facilitating evolutionary innovation by developmental modularity and variability. In: Generative & Developmental Systems Workshop at 18th Genetic & Evolutionary Computation Conference, Montreal, Canada (July 2009)
3. Dyson, S., Gurdon, J.B.: The interpretation of position in a morphogen gradient as revealed by occupancy of activin receptors. Cell 93, 557–568 (1998)
4. Eggenberger, P.: Evolving morphologies of simulated 3d organisms based on differential gene expression. In: Husbands, P., Harvey, I. (eds.) Proc. 4th European Conference on Artificial Life (1997)
5. Federici, D., Downing, K.: Evolution and development of a multicellular organism: Scalability, resilience, and neutral complexification. Artificial Life 12, 381–409 (2006)
6. Gilcrease, M.Z.: Integrin signaling in epithelial cells. Cancer Lett. 247, 1–25 (2007)
7. Jiang, J., Levine, M.: Binding affinities and cooperative interactions with bHLH activators delimit threshold responses to the dorsal gradient morphogen. Cell 72, 741–752 (1993)
8. Klavins, E., Ghrist, R., Lipsky, D.: A grammatical approach to self-organizing robotic systems. IEEE Trans. Autom. Control 51(6), 949–962 (2006)
9. Lander, A.D.: Morpheus unbound: Reimagining the morphogen gradient. Cell 128, 245–256 (2007)
10. Lander, A.D., Nie, Q., Wan, F.Y.M.: Do morphogen gradients arise by diffusion? Developmental Cell 2, 785–796 (2002)
11. Lawrence, P.A.: The Making of a Fly. Blackwell Science Ltd., Malden (1992)
12. Martz, E., Steinberg, M.S.: The role of cell-cell contact in "contact" inhibition of cell division: A review and new evidence. J. Cell. Physiol. 79, 189–210 (1971)
13. Nagpal, R.: Programmable Self-Assembly: Constructing Global Shape Using Biologically-Inspired Local Interactions and Origami Mathematics. Ph.D. thesis, Massachusetts Institute of Technology (2001)
14. Spirov, A., Fahmy, K., Schneider, M., Frei, E., Noll, M., Baumgartner, S.: Formation of the *bicoid* morphogen gradient: an mRNA gradient dictates the protein gradient. Develoment 136, 605–614 (2009)
15. Strutt, D.: Organ shape: Controlling oriented cell division. Current Biology 15, R758–R759 (2005)
16. Werfel, J.: Anthills Built to Order: Automating Construction with Artificial Swarms. Ph.D. thesis, Massachusetts Institute of Technology (2006)
17. Wolpert, L.: Positional information and the spatial pattern of cellular differentiation. J. Theol. Biol. 25(1), 1–47 (1969)
18. Yim, M., Shen, W.M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.S.: Modular self-reconfigurable robot systems. IEEE Robotics & Automation Magazine, 2–11 (March 2007)

Evaluating the Robustness of Activator-Inhibitor Models for Cluster Head Computation

Lidia Yamamoto[1] and Daniele Miorandi[2]

[1] Data Mining and Theoretical Bioinformatics Team (FDBT)
Image Sciences, Computer Sciences and Remote Sensing Laboratory (LSIIT)
University of Strasbourg, France
lidia.yamamoto@unistra.fr
[2] Pervasive Team, CREATE-NET, Trento, Italy
daniele.miorandi@create-net.org

Abstract. Activator-inhibitor models have been widely used to explain several morphogenetic processes. They have also been used to engineer algorithms for computer graphics, distributed systems and networks. These models are known to be robust to perturbations such as the removal of peaks of chemicals. However little has been reported about their actual quantitative performance under such disruptions.

In this paper we experimentally evaluate the robustness of two well-known activator-inhibitor models in the presence of attacks that remove existing activator peaks. We focus on spot patterns used as distributed models for cluster head computation, and on their potential implementation in chemical computing. For this purpose we derive the corresponding chemical reactions, and simulate the system deterministically.

Our results show that there is a trade-off between both models. The chemical form of the first one, the Gierer-Meinhardt model, is slow to recover due to the depletion of a required catalyst. The second one, the Activator-Substrate model, recovers more quickly but is also more dynamic as peaks may slowly move. We discuss the implications of these findings when engineering algorithms based on morphogenetic models.

1 Introduction

An activator-inhibitor model is a special case of a reaction-diffusion system where two chemicals interact in an antagonistic way, resulting in Turing patterns in space [20], such as spots and stripes on the skin of animals (e.g. leopard, zebra).

Activator-inhibitor models are recurrent components in morphogenesis. They offer an abstract model to explain many different morphogenetic phenomena, including the regular spacing of cactus thorns and bird feathers, shape regeneration after damage, the production of sequences of repeated elements such as insect body segments, the assembly of photoreceptor cells in insect eyes, and the positioning of leaves in growing plants [4,15,16,6,13]. They have also been used as inspiration for algorithms to produce textures and landscapes in computer graphics, and for autonomous, decentralized, distributed coordination algorithms, for

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 143–154, 2010.

instance in amorphous computing [1], wireless and sensor networks [10,17,14], and autonomous surveillance systems [21,12].

It has been pointed out that activator-inhibitor models are robust to perturbations in the resulting patterns [15]. For instance, removing an activator peak results in this peak being regenerated or another peak being formed at a nearby place. However little has been reported about the actual quantitative performance of activator-inhibitor models under perturbations.

In this paper we experimentally evaluate the robustness of two well-known activator-inhibitor models in the presence of attacks that remove existing activator peaks. The first model is the well-known Gierer-Meinhardt Activator-Inhibitor Model and the second one is the Activator-Depleted Substrate Model. Both models are extensively described in [15,13].

We focus on spot patterns (as opposed to stripes). Spots can be used to place differentiated functions at activator peaks, such as cluster heads in ad hoc and sensor networks [22,11,19]. Cluster heads can serve a variety of functions, such as aggregating information from nearby sensor nodes, or saving energy by switching off redundant nodes. The activator-inhibitor system used in this way can be regarded as a distributed algorithm that solves an instance of a cluster head election problem.

Instead of the traditional approach of directly integrating the reaction-diffusion equations provided, we derive the corresponding equations from chemical reactions using the law of mass action. The goal of this method is to ensure compatibility with a potential chemical implementation, either in an artificial chemistry [8,7], or in natural chemical computing such as reaction-diffusion processors [2]. For this purpose, we reverse engineer both models from the equations back to the chemical reactions, and then simulate the resulting system deterministically by integrating the new set of equations obtained.

Our results show that the chemical derivation of the first model, the Gierer-Meinhardt model, seems not so robust as expected, since peaks of activator do not easily recover due to the depletion of a required catalyst. This catalyst does not appear in the original equations, but is necessary to achieve the desired inhibition effect. The second model evaluated, the Activator-Substrate model, is much faster at recovering from perturbations on activator peaks. On the other hand, it is also more dynamic, and peaks tend to change location sometimes. Therefore, there is a trade-off between both systems. We discuss the implications of these findings for engineering algorithms based on morphogenetic mechanisms.

2 Chemical Kinetics and Reaction-Diffusion Systems

The *Law of Mass Action* states that, in a well-stirred reactor, the average speed (or rate) of a chemical reaction is proportional to the product of the concentrations of its reactants [3].

The concentration dynamics of all molecules in the system can be described by a system of ordinary differential equations (ODE), where each equation describes the change in concentration of one particular molecular species. This ODE system can be expressed in matrix notation as follows:

$$\frac{d\boldsymbol{c}}{dt} = \mathbf{M}\boldsymbol{v} \tag{1}$$

where $d\boldsymbol{c}/dt$ is the vector of differential equations for each of the species C_i (with concentration c_i), in the system; M is the stoichiometric matrix of the system; and $\boldsymbol{v}$ is a vector of rates for each reaction. The rates may follow the law of mass action as well as other laws such as enzyme and Hill kinetics.

A reaction-diffusion system is a chemical reaction system in which substances not only react but may also diffuse in space. It is expressed by a system of partial differential equations (PDE) describing the change in concentrations of substances caused by both reaction and diffusion effects combined:

$$\frac{\partial \mathbf{c}}{\partial t} = \mathbf{f}(\boldsymbol{c}) + D\nabla^2\boldsymbol{c} \tag{2}$$

The vector $\boldsymbol{c}$ now refers to the concentration level c_i of each chemical C_i at position (x, y) in space. The reaction term $\mathbf{f}(\boldsymbol{c})$ describes the reaction kinetics (like in Eq. (1), but now expressed for each point in space). The diffusion term $D\nabla^2\boldsymbol{c}$ tells how fast each chemical substance will diffuse in space. D is a diagonal matrix containing the diffusion coefficients, and ∇^2 is the Laplacian operator.

3 Activator-Inhibitor Models

Activation-inhibition models describe situations in which two competing processes take place over space and time. The first one (*short-range activation*) tends to self-enhance the process within the local neighbourhood. A competing force (*long-range inhibition*), weaker but with a longer spatial range, tends to decrease the activation effect in the surrounding space. Under certain conditions, the interaction between short-range activation and long-range inhibition can lead to the formation of asymmetric spatial patterns, such as spots and stripes resembling those found on the skin of animals. Alan Turing [20] was the first to present a mathematical model of morphogenesis based on reaction-diffusion dynamics, including activator-inhibitor dynamics.

In chemistry, activators and inhibitors are molecules that may diffuse over space. Activators trigger autocatalytic reactions that increase their own concentration (self-enhancement), but such effect has limited spatial range. On the other hand, activators also trigger inhibitory reactions that cause their own concentrations to decrease. Such a "negative impact" process has a reduced intensity when compared to self-enhancement, but has much larger spatial range.

Numerous alternative activator-inhibitor models are available in literature [4,20,15,16,13]. They all achieve similar short-range activation and long-range inhibition effects, but differ in the chemicals needed, the way they interact, and the characteristics of the resulting patterns. In this paper we focus on two of these models: The Gierer-Meinhardt Activator-Inhibitor Model, and the Activator-Depleted Substrate Model, both from [15]. We describe them below.

3.1 The Gierer-Meinhardt Activator-Inhibitor Model

One of the most widely used activator-inhibitor model is named after Gierer and Meinhardt [15], and is described by the following reaction-diffusion equations:

$$\frac{\partial a}{\partial t} = \frac{\sigma a^2}{h} - \mu_a a + \rho_a + D_a \nabla^2 a \tag{3}$$

$$\frac{\partial h}{\partial t} = \sigma a^2 - \mu_h h + \rho_h + D_h \nabla^2 h \tag{4}$$

In this model, a represents the concentration of a short–range autocatalytic substance (activator A) and h the concentration of is its long-range antagonist H, the inhibitor.

The concentration a tends to self-enhance (growth proportional to σa^2), but such growth is slowed down by the inhibitor by a factor $1/h$. The inhibitor is produced by the molecular collision of two activator molecules, and this reaction contributes to its growth by a factor of σa^2. Further, both activator and inhibitor concentrations decay proportionally to their respective values. The constants μ_a and μ_h describe the rates at which each substance decays. The terms ρ_a and ρ_h represent a constant inflow of substances A and H respectively. Molecules can move across nearby cells following the concentration gradient, hence the diffusion terms $D_a \nabla^2 a$ and $D_h \nabla^2 h$, where D_a and D_h are constant diffusion coefficients.

An important condition for the formation of asymmetric patterns in this model is $D_h \gg D_a$ (the inhibitor diffuses much faster than the activator). Another condition is that $\mu_h > \mu_a$ (the inhibitor drains more quickly than the activator). A full mathematical description of the necessary and sufficient conditions for pattern formation in activator-inhibitor systems can be found in [16].

Figure 1 (left) illustrates the typical pattern formation process resulting from this activator-inhibitor model: starting from a homogeneous mix of chemicals that is slightly perturbed, the system progressively self-organizes into spot patterns, where the spots are regions of high activator concentration.

This model is widespread in literature [4,20,15,16,13], but little has been discussed about its actual chemical implementation, either in natural or artificial chemistries. This is important if one would like to engineer reaction-diffusion

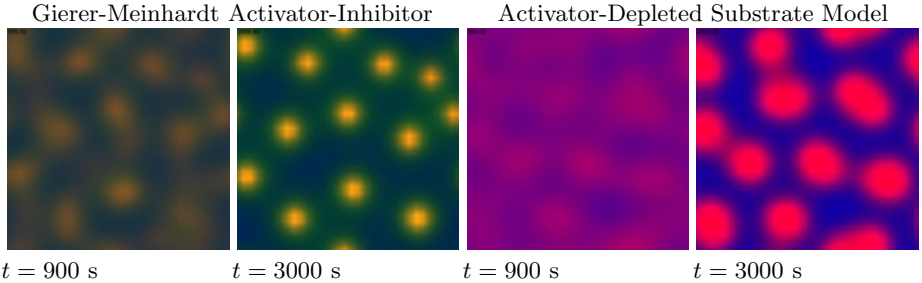


Fig. 1. Typical patterns resulting from the two activator-inhibitor models studied

systems using reaction-diffusion computers [2], or by evolving the corresponding chemical reaction graphs [5].

What set of chemical reactions can lead to equations (3) and (4)? Especially, the term $1/h$ in Eq. (3) does not seem to stem directly from mass action kinetics, neither from other well-known kinetic laws such as enzyme kinetics or Hill kinetics. In [4] it has been pointed out that the term $1/h$ comes from the effect of a third substance, a catalyst C, which is assumed to be in steady state.

Whereas an ODE system can be directly constructed from a system of chemical reactions, currently there is no firm and generic method to do the reverse operation, i.e. to derive the corresponding chemical reactions from a given ODE. We have reverse-engineered equations (3) and (4) using the catalyst hint from [4], and obtained the following result:

$$2A \xrightarrow{\sigma} 2A + H \quad (5) \qquad C + 2A \xrightarrow{k_1} C + 3A \quad (10)$$

$$A \xrightarrow{\mu_a} \emptyset \quad (6) \qquad C + H \xrightarrow{k_2} H \quad (11)$$

$$H \xrightarrow{\mu_h} \emptyset \quad (7) \qquad C \xrightarrow{\mu_c} \emptyset \quad (12)$$

$$\emptyset \xrightarrow{\rho_a} A \quad (8) \qquad \emptyset \xrightarrow{\rho_c} C \quad (13)$$

$$\emptyset \xrightarrow{\rho_h} H \quad (9)$$

Note that the first five reactions (5)-(9) stem directly from equations (3) and (4), while the other reactions (10)-(13) involve a "hidden" chemical C. C acts as a catalyst in the production of the activator A, however the inhibitor H consumes C, and it is the depletion of C through H that causes the inhibitory effect on A. This can be shown via the following analysis. We start by deriving the differential equations corresponding to the full set of reactions (5)-(13), using the Law of Mass Action:

$$\frac{da}{dt} = k_1 c a^2 - \mu_a a + \rho_a \quad (14)$$

$$\frac{dh}{dt} = \sigma a^2 - \mu_h h + \rho_h \quad (15)$$

$$\frac{dc}{dt} = -k_2 c h - \mu_c c + \rho_c \quad (16)$$

Eq. (15) corresponds directly to the reaction part of Eq. (4), so we do not need to look at it further. Now we focus on how to obtain the reaction part of Eq. (3) from Eqs. (14) and (16). For simplification we assume that $\mu_c = 0$, so that the inhibitor is the only responsible for a depletion of C. At steady state, the concentration of C stays stable, thus:

$$\frac{dc}{dt} = -k_2 c h + \rho_c = 0 \Rightarrow c = \frac{\rho_c}{k_2 h} \quad (17)$$

Substituting Eq. (17) in (14) we obtain:

$$\frac{da}{dt} = \frac{k_1 \rho_c a^2}{k_2 h} - \mu_a a + \rho_a \quad (18)$$

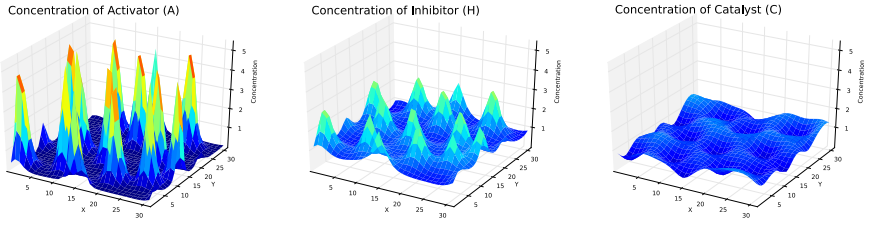


Fig. 2. Concentration levels of activator (left), inhibitor (center), and catalyst (right) for the Activator-Inhibitor pattern on Figure 1 (left) at $t = 3000$s

By setting $\sigma = k_1\rho_c/k_2$ we obtain the reaction part of Eq. (3) as needed. Note therefore that Eq. (3) is only accurate when the system is in steady state. Perturbations may cause the catalyst concentrations to fluctuate, and therefore the approximation in Eq. (3) becomes a poor modelling of the system's dynamic behavior in these cases. This might partly account for the performance issues uncovered in our simulation experiments reported in Section 5.

Figure 2 shows the concentrations of activator, inhibitor and catalyst at the end of the simulation that generated Fig. 1 (left). We can see how the activator and inhibitor peaks coincide, the inhibitor peaks being more modest, while the catalyst valleys correspond to regions of high activator concentration.

3.2 Activator-Depleted Substrate Model

Several variations over the basic activator-inhibitor model have been proposed in the mathematical biology literature [9,15,16], some encompassing a larger number of equations, representing situations in which more complex interactions among molecules take place.

An interesting alternative approach in our context is the *activator-depleted substrate model* [15], or activator-substrate for short. Instead of modelling the explicit presence of a molecular species able to slow down the activation process, the activator-substrate model achieves a similar antagonistic effect through the depletion of a substrate S, which gets consumed during the production of the activator A. The resulting reaction-diffusion equations read:

$$\frac{\partial a}{\partial t} = \sigma_a s a^2 - \mu_a a + \rho_a + D_a \nabla^2 a \tag{19}$$

$$\frac{\partial s}{\partial t} = -\sigma_s s a^2 - \mu_s s + \rho_s + D_s \nabla^2 s \tag{20}$$

The substrate S gets consumed during the autocatalytic production of activator A. Both activator and substrate are produced everywhere at constant rate ρ_a and ρ_s respectively, and decay at rate μ_a (resp. μ_s). Figure 1 (right) shows the typical pattern formation process using this model. Note that, in contrast to the previous model, here spots slowly change, and sometimes grow and divide, i.e.

they can replicate. This behavior has been thoroughly studied in the Gray-Scott model [18], which is a special case of activator-substrate model.

Note that if it was not for the possibility to have different σ values for A and S (σ_a and σ_s), then it would have been straightforward to derive the corresponding chemical reactions. In order to allow for different σ_a and σ_s, we have come up with the following solution:

$$S + 2A \xrightarrow{\sigma_a} 3A \quad (21) \qquad \emptyset \xrightarrow{\rho_s} S \quad (24)$$

$$S + 2A \xrightarrow{k_s} 2A \quad (22) \qquad A \xrightarrow{\mu_a} \emptyset \quad (25)$$

$$\emptyset \xrightarrow{\rho_a} A \quad (23) \qquad S \xrightarrow{\mu_s} \emptyset \quad (26)$$

where $k_s = \sigma_s - \sigma_a$. With this we obtain the reaction part of Eqs. (19) and (20) from the law of mass action. Reactions (21) and (22) imply that, when two molecules of A and one of S react together, one extra molecule of A may be formed in some cases (reaction (21)), while in others (reaction (22)) the substrate molecule will simply be lost or degraded into something that can not be used by the system. So the second reaction can be seen as a kind of "error" or inefficiency in the process of autocatalysis of A, which can be nevertheless exploited to construct useful patterns.

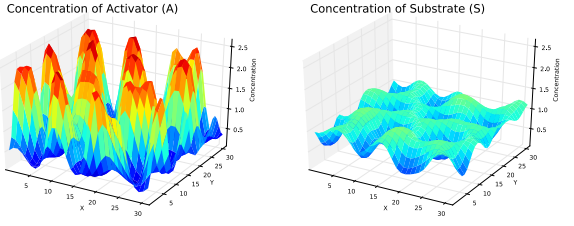


Fig. 3. Concentration levels of activator (left) and substrate (right) for the Activator-Substrate pattern on Figure 1 (right) at $t = 3000$s

Figure 3 shows the concentrations of activator and substrate at the end of the simulation that generated Fig. 1 (right). The substrate valleys correspond to regions of high activator concentration. Note the wider and smoother activator peaks when compared to Fig. 2.

4 Experimental Setup

We compare the two models described in the previous section: the original Gierer-Meinhardt model (henceforth referred to as simply "activator-inhibitor"), and the Activator-Depleted Substrate model (referred to as simply "activator-substrate"). We simulate a chemical implementation of both reaction-diffusion systems, according to their corresponding chemical reactions (5)-(13) and (21)-(26). This is in contrast to the traditional approach of integrating equations (3), (4), (19) and (20) directly.

Table 1. Parameters used in the experiments

Activator-Inhibitor			Activator-Substrate		
$\sigma = 0.02$	$\mu_a = 0.01$	$\rho_a = 0$	$\sigma_a = 0.01$	$\mu_a = 0.01$	$\rho_a = 0$
$k_1 = 0.01$	$\mu_h = 0.02$	$\rho_h = 0$	$\sigma_s = 0.02$	$\mu_s = 0$	$\rho_s = 0.02$
$k_2 = 0.1$	$\mu_c = 0.1$	$\rho_c = 0.1$	$D_a = 0.008$	$D_s = 0.2$	$k_s = 0.01$
$D_a = 0.005$	$D_h = 0.2$				

We specify the chemical reactions, together with their rates, in a text format such as "`A + 2 B --> C , k=0.1`". Each line expresses a chemical reaction. The lines are then parsed, and the corresponding stoichiometric matrices are constructed, together with their reactant and product multisets. This information is then used to drive a generic integrator (based on the simple explicit Euler method) automatically, according to equations (1) and (2). This system is intuitive and versatile, and can simulate any kind of reaction-diffusion system with little effort. Besides ensuring chemical compatibility, another interest of such system is to make sure that the formula and behavior obtained stem directly from the chemical reactions, without hidden assumptions or simplifications. The visualization is performed with the help of the Breve simulator[1].

The parameters of the activator-inhibitor and activator-substrate models can be found in Table 1. Except for the extra coefficients k_i, all other constants are set to the same values as [13].

We choose the diffusion coefficients D_a for the activator such that the average number of peaks obtained, and their average spacing, is approximately the same in both models, such that they can be compared.

We simulate both systems on a regular grid of 32x32 cells, using an integration timestep of $dt = 0.1$. Two scenarios are tested for each model: with and without perturbation. In the scenario without perturbation, the simulation runs without interference until it finishes at 10K simulation seconds. In the perturbed scenario, at $t = 4000s$, after the system has reached a stable pattern, perturbation events are are introduced every 1000 seconds. Each perturbation event happens as follows: 50% of the existing peaks are selected at random for disruption. If selected, a peak loses 90% of its activator concentration, and the same percentage is removed from the Moore neighborhood (8 surrounding cells) around the peak. This simulates severe disruptions, such those caused by mechanically removing chemicals, or by damaging patches of processors in amorphous computers.

A peak is a point with maximum activator concentration, i.e. the local concentration at location (x, y) is higher than all the concentrations on the Von Neumann neighborhood (north-south-east-west cells) around the peak. Moreover the concentration at the peak must be above a threshold for it to be considered as a cluster head. In this evaluation study we consider only the peaks elected as cluster heads, therefore we use the terms peak and cluster head interchangeably. The peak concentration threshold is $a_{min} = 3.0$ for the activator-inhibitor model, and $a_{min} = 2.0$ for the activator substrate model.

[1] `www.spiderland.org`

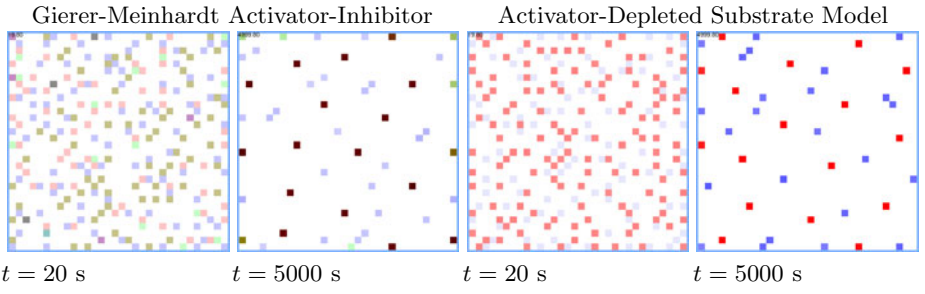


Fig. 4. Cluster head election example

Figure 4 shows an example of the process of electing activator peaks as cluster heads using the algorithm. At the beginning of the simulation ($t = 20$s) there are only very low peaks (represented by the light shaded spots) which are not elected because they are below the threshold. At $t = 5000$ seconds we can clearly see the elected peaks, indicated by brown spots in the first model, and red spots in the second. The blue spots indicate catalyst and substrate peaks which play no function as cluster heads.

5 Results

The performance of each algorithm is measured by the average distance between nearest peaks, the total number of elected peaks, and the number of cells that are served by a peak. The average distance tells the average spacing among peaks, and the number of cells that are served measures the number of cells that are located at a distance shorter than a threshold of the nearest peak, i.e. they can receive a service from this peak, such as the aggregation of some sensor information. We calculate the average distance between nearest peaks by looking at the four nearest peaks of each peak, and calculating the average of this distance for all peaks.

Figure 5 shows the average distance between nearest peaks (errorbars indicate the standard deviation). We can see that the distance remains stable when there are no disruptions. Under perturbations, the activator-inhibitor model is less stable, presenting bigger fluctuations than the activator-substrate model. Note however, that while the distance in the activator-inhibitor model remains totally constant in the absence of perturbations, in the activator-substrate model this distance fluctuates slightly, even in the absence of perturbations. This indicates that peaks are constantly moving a bit, as the underlying spots merge, grow and divide. This can be more easily seen in Figure 6 which shows the number of peaks elected as cluster heads. Indeed, this behavior can be easily observed when watching an animation of the activator-substrate system.

Figure 7 shows the number of cells that are at a distance bigger than $d_{max} = 10$ cells, such that they do not receive service from any nearby peak. Here we can

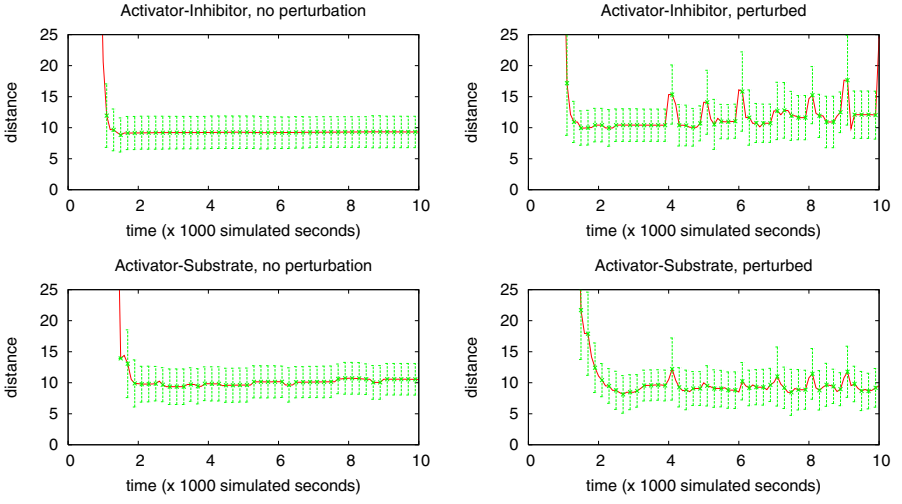


Fig. 5. Average distance between nearest peaks

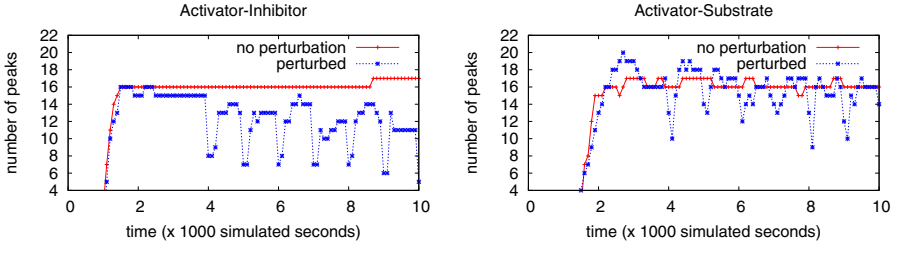


Fig. 6. Number of peaks elected as cluster heads

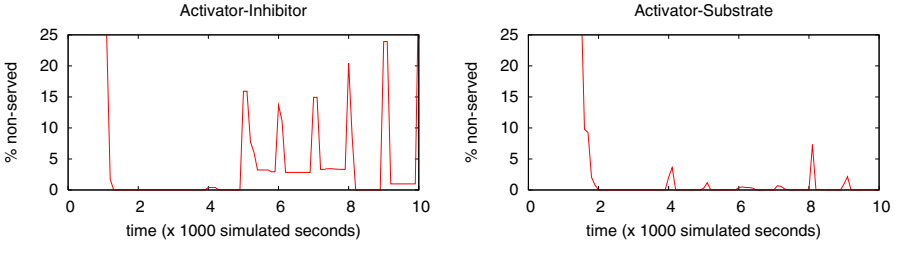


Fig. 7. Percentage of non-served cells in both models, under perturbation

clearly see that, under perturbation, the activator-inhibitor model leaves a larger number of cells without service than the activator-substrate model, although the average peak spacing (according to Fig. 5) is similar in both cases.

Note that, because it involves less chemicals and less reactions, the activator-substrate model is faster to compute: the simulations of the activator-inhibitor model take about 30% longer.

6 Conclusions

Activation-inhibition models have been used as models of distributed computation, to solve coordination problems such as the placement of cluster heads and the formation of content delivery highways. However, their parametrization is not simple, and performance trade-offs have to be faced.

The results reported in this paper can serve as guidelines in the design of robust algorithms based on spot patterns obtained via activation-inhibition. Two variants were evaluated, one based on an inhibitor that consumes a catalyst necessary for the activator to grow, and another one based on the depletion of a substrate also needed for activator growth.

Our results show that there is a trade-off between the stability but potentially slow response to perturbations in the activator-inhibitor model, and on the other hand, the fast response to perturbations at the cost of slowly "moving peaks" in the activator-substrate model. Therefore the first model is more appropriate in situations where, once the pattern is formed, it is not supposed to change often. While the latter is useful in more dynamic situations where frequent changes are expected, and the system should constantly react to them by quickly reconfiguring itself to a new valid configuration.

One might ask whether our results could be mere artifacts of our particular chemical implementations of both models. There might be other solutions leading to sets of reactions that do not suffer from the catalyst depletion problem found in our chemical solution for the Gierer-Meinhardt model. And there might be other solutions to the activator-substrate, or other parameter ranges where the peaks are more static. Nevertheless, we believe that our results are fairly representative of the general characteristics of both models.

As future work, a stochastic simulation of the system in larger and irregular amorphous topologies is necessary. Moreover, it would be useful to evaluate other reaction-diffusion approaches and their robustness when used as an engineering tool. It would also be interesting to study their behavior and performance in combination with other morphogenetic mechanisms. Another interesting topic for future work is to investigate the automatic evolution of reaction networks leading to the desired patterns.

Acknowledgments. This work was supported by the European Union through the BIONETS Project EU-IST-FET-SAC-FP6-027748, `www.bionets.eu`. It was performed when the first author was with the University of Basel, Switzerland. The authors would like to thank David Lowe (University of Technology Sydney, Australia) for the enticing discussions that motivated this work.

References

1. Abelson, H., et al.: Amorphous computing. Communications of the ACM 43 (2000)
2. Adamatzky, A., Costello, B.D.L., Asai, T.: Reaction-Diffusion Computers. Elsevier Science Inc., New York (2005)
3. Atkins, P., de Paula, J.: Physical Chemistry. Oxford University Press, Oxford (2002)
4. Bar-Yam, Y.: Dynamics of Complex Systems. Westview Press (2003)
5. Deckard, A., Sauro, H.M.: Preliminary Studies on the In Silico Evolution of Biochemical Networks. ChemBioChem 5(10), 1423–1431 (2004)
6. Deutsch, A., Dormann, S.: Cellular automaton modeling of biological pattern formation: characterization, applications, and analysis. Birkhäuser, Basel (2005)
7. Dittrich, P.: Chemical Computing. In: Banâtre, J.-P., Fradet, P., Giavitto, J.-L., Michel, O. (eds.) UPP 2004. LNCS, vol. 3566, pp. 19–32. Springer, Heidelberg (2005)
8. Dittrich, P., Ziegler, J., Banzhaf, W.: Artificial Chemistries – A Review. Artificial Life 7(3), 225–275 (2001)
9. Dormann, S.: Pattern Formation in Cellular Automaton Models. Ph.D. thesis, University of Osnabrück, Dept. of Mathematics/Computer Science (2000)
10. Durvy, M., Thiran, P.: Reaction-diffusion based transmission patterns for ad hoc networks. In: INFOCOM, pp. 2195–2205 (2005)
11. Erciyes, K., et al.: Graph theoretic clustering algorithms in mobile ad hoc networks and wireless sensor networks. Appl. Comput. Math. 6, 162–180 (2007)
12. Hyodo, K., Wakamiya, N., Murata, M.: Reaction-diffusion based autonomous control of camera sensor networks. In: Proc. Bionetics, Budapest, Hungary (2007)
13. Koch, A.J., Meinhardt, H.: Biological pattern formation: from basic mechanisms to complex structures. Reviews of Modern Physics 66(4) (1994)
14. Lowe, D., Miorandi, D., Gomez, K.: Activation-inhibition-based data highways for wireless sensor networks. In: Proc. Bionetics. ICST, Avignon (2009)
15. Meinhardt, H.: Models of biological pattern formation. Academic Press, London (1982)
16. Murray, J.D.: Mathematical Biology: Spatial models and biomedical applications. Mathematical Biology, vol. 2. Springer, Heidelberg (2003)
17. Neglia, G., Reina, G.: Evaluating activator-inhibitor mechanisms for sensors coordination. In: Proc. Bionetics. ICST, Budapest (2007)
18. Pearson, J.E.: Complex patterns in a simple system. Science 261(5118), 189–192 (1993)
19. Soro, S., Heinzelman, W.B.: Cluster Head Election Techniques for Coverage Preservation in Wireless Sensor Networks. Ad Hoc Networks 7, 955–972 (2009)
20. Turing, A.M.: The chemical basis of morphogenesis. Philosophical Transactions of the Royal Society of London B 327, 37–72 (1952)
21. Yoshida, A., Aoki, K., Araki, S.: Cooperative control based on reaction-diffusion equation for surveillance system. In: Khosla, R., Howlett, R.J., Jain, L.C. (eds.) KES 2005. LNCS (LNAI), vol. 3683, pp. 533–539. Springer, Heidelberg (2005)
22. Yu, J.Y., Chong, P.H.J.: A survey of clustering schemes for mobile ad hoc networks. IEEE Communications Surveys and Tutorials 7, 32–48 (2005)

Evolution of Self-organised Path Formation in a Swarm of Robots

Valerio Sperati, Vito Trianni, and Stefano Nolfi

Istituto di Scienze e Tecnologie della Cognizione,
Consiglio Nazionale delle Ricerche, Rome, Italy
{valerio.sperati,vito.trianni,stefano.nolfi}@istc.cnr.it

Abstract. We present a set of experiments in which a robotic swarm manages to collectively explore the environment, forming a path to navigate between two target areas, which are too distant to be perceived by an agent at the same time. Robots within the path continuously move back and forth between the two locations, exploiting visual interactions with their neighbours. The global group behaviour is obtained through an evolutionary process and presents emergent properties like robustness, path optimisation and scalability, which recall ants trail formation.

1 Introduction

Exploration and navigation in unknown environments represent basic activities for most animal species, and efficient strategies can make the difference between death and survival. For this reason, Nature presents a wide range of possibilities, each particularly adapted to the task to be accomplished and to the sensory-motor and cognitive abilities of the species under observation. In primates, as well as in other animals, navigation abilities are usually linked to mental representations of the environment, referred to as "cognitive maps". For instance, it has been found that specific neurons of the rodents hippocampus (called "place cells") have a high firing rate in correspondence of specific locations in the environment [15]. Neural representations seem to characterise also the behaviour of insects. A map-like organisation of spatial memory has been proposed for honeybees, which are able to retrieve the navigation path on the basis of learned landmarks around the hive [12]. A similar strategy is employed also by the desert ants of the genus *Cataglyphis*, which however couple the landmark-base strategy with their skylight (polarization) compass and path integrator (ants integrate over time the path covered through a sort of vector summation) that allow them to return to the nest following a straight line [21]. Ant species that forage in groups rely on a collective strategy for exploration and navigation, exploiting the well known mechanism of pheromone trail formation: when moving from a foraging patch to the nest, ants lay a blend of pheromones that can be exploited by other ants to reach the same patch. Thanks to this strategy, ants can efficiently navigate in the environment and optimise the path between nest and food [7,2].

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 155–166, 2010.

In Robotics too, much attention has been paid to the navigation and exploration problems, and several different strategies have been proposed. Map-based navigation exploits probabilistic approaches to solve the so called simultaneous localisation and mapping (SLAM) problem [18,1], as well as biologically inspired ones [4,9,6]. Similarly, landmark-based navigation and path integration have been exploited, often with a close look at biology [23,8,10,20]. For what concerns collective strategies inspired to the ants trails, however, research has to confront with the complex problem of finding an alternative to the pheromones, given that chemical substances are difficult to exploit in a robotic setup. Instead of chemicals, the most common approach is to rely on communication and message passing among robots, which therefore simulate pheromone attributes on a communication network [17]. Other approaches exploit the robots themselves as markers for the trail: marker robots remain static and signal a path between two locations in the environment, while explorer robots exploit this path to efficiently navigate [3,22,16,14].

In this paper, we study the abilities of a robotic swarm to explore an environment and coordinately navigate between two target areas by exploiting a simple form of visual communication. We start from two basic assumptions about the usage of communication signals. On the one hand, communication can be exploited to signal the position of a target zone to the other robots, therefore facilitating the exploration task. For this purpose, robots are provided with a red LED positioned on their back, which has the same colour of the target area and therefore elicits the same effect of the visual perception of a target area [11]. On the other hand, communication can be used to coordinate the movements of the robots in the environment, therefore supporting the navigation task. For this purpose, robots are provided with a blue LED on their front, which can signal their position and heading to other robots.

By exploiting evolutionary robotics techniques [13,5], we study the emergence of behavioural and communication strategies. Evolutionary robotics is particularly useful to synthesise self-organising collective behaviours, characterised by properties such as robustness, flexibility and scalability [19]. We analyse the system trying to identify the relevant properties of the evolved behaviours for what concerns both the individual rules followed by the robots, and the communication signals exploited. Despite not explicitly required by the fitness function, the evolved behaviour presents features that are similar to the trail formation in ants: robots form a trail between the target areas and robustly maintain it, also optimising the shape towards the shortest path (see Sec. 3). Moreover, we analyse the generalisation and scalability property of the collective behaviour, by testing it in different environmental conditions and with larger swarms (see Sec. 4). Discussions are reported in Sec. 5.

2 Experimental Setup

The experimental scenario involves a swarm of wheeled robots, whose behaviours have been evolved for the ability to navigate back and forth as quickly as possible

between two target areas, located within an arena surrounded by walls (Fig. 1). Since target areas can be perceived only from a short distance, the robots should be able to find them by exploring the environment. Moreover, in order to quickly navigate from one target area to the other without relying on time-consuming exploratory behaviours, the robots should be able to preserve, in some way, some information concerning the location of previously visited areas. Since the robots are rewarded on the basis of the efficiency with which each individual is able to accomplish the task (see eq. (3)), the evolutionary process might potentially lead to the development of non-cooperative solutions in which the fact to be part of a swarm does not provide any advantage. However, as we will see, the evolutionary process rather leads to strategies in which the robots coordinate and cooperate to find the target areas. In particular, the problem of preserving a trace of the position of previously visited areas is solved by generating and maintaining a dynamic path that connects the two target areas, which allows the individuals of the swarm to efficiently navigate between them. In this section, we detail the experimental setup and the evolutionary algorithm used.

2.1 The Robots and the Environment

Ten simulated robots are placed in a rectangular arena (height $H = 250$ cm; width variable within the interval $W \in [250, 290]$ cm). The target areas consist in two circles painted in grey (diameter $d = 32$ cm). A target area can be perceived from distance by the robot thanks to a red LED placed over its centre. This red LED is indistinguishable from the one provided to the robots (see below).

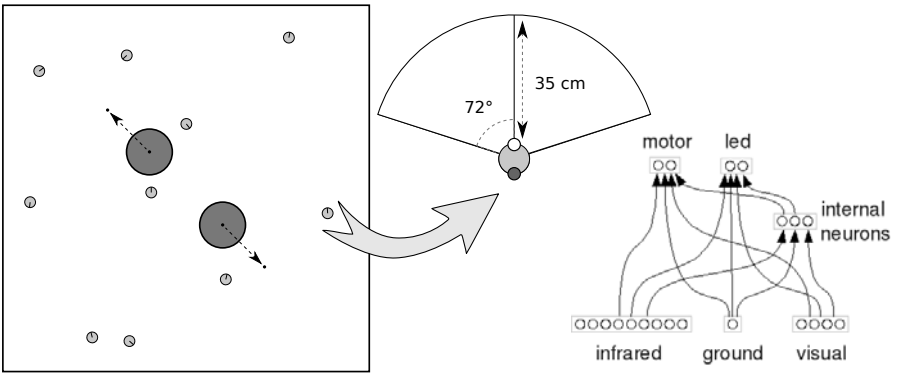


Fig. 1. The experimental setup. On the left, a snapshot of the simulated environment is shown. Ten robots randomly positioned in the environment are represented as small circles. The grey disks represent the circular target areas with a red LED in the centre. The distance between the area centres is $D = 70$ cm, while the arrows indicate which is their maximum displacement, when $D = 150$ cm. In the centre, we show a schematic representation of the robot's vision sensors, indicating the sectors and the perceptual range.The blue and red LED position is indicated as a white and a gray dot, respectively. On the right, the architecture of the robots' neural controller is shown.

The two areas are positioned symmetrically with respect to the centre of the arena at a fixed distance D (see Fig. 1). Each robot is provided with two motors that control two wheels, providing a differential drive motion (maximum speed: $v_{max} = 8.2$ cm/s). Moreover each robot is provided with a blue LED on the front and a red LED on the rear of its body. Both can be switched on and off by the robot controller. Additionally, a robot is provided with: (i) 8 infrared sensors uniformly distributed around the robot body, used to detect obstacles or other robots up to a distance of about 2.5 cm; (ii) 1 ground sensor located under the front of the robot, used to detect whether the robot is placed over a target area or not; and (iii) 4 vision sensors, used to detect the presence of red or blue LEDs (2 sensors for each colour). The vision sensors return a binary value about the presence or absence of LEDs in two 72° sectors of the image that cover the front-left and front-right area of the robot. Both red and blue LEDs can be detected up to a distance of 35 cm (see Fig. 1).

2.2 The Controller and the Evolutionary Algorithm

Each robot is controlled by a feed-forward neural network (see Fig. 1) with 13 sensory neurons (8 infrared, 1 ground, 4 vision sensors), 3 internal leaky integrators neurons and 4 motor neurons (2 wheels, 1 blue LED and 1 red LED). Connections weights, biases and time constants of the leaky integrators are genetically encoded parameters subject to artificial evolution [13,5]. The free parameters of the robot's neural controller are encoded in a binary genotype, using 8 bits for each real number. The connection weights and biases can vary in the range $[-5, 5]$, while the time constants vary in $[0, 1]$. Evolution works on a population of 100 randomly generated genotypes. After evaluation of the fitness, the 20 best genotypes survive in the next generation (elitism), and reproduce by generating four copies of their genes with a 3% mutation probability of flipping each bit. The evolutionary process lasts 500 generations.

In order to evaluate the fitness, a genotype is translated into N identical neural controllers which are downloaded onto N identical robots (i.e., the group is homogeneous). Each group of robots is tested for 15 trials, each lasting 6000 time-steps (one time-step corresponds to 100 ms). Regarding the fitness computation, the trial is split into two periods, T_{add} and T_{eff}: the latter lasts 5400 time-steps and is the actual period during which the fitness is estimated. The former lasts 600 time-steps, and is the time dedicated to the exploration of the environment and to the initial coordination of the robots. At the beginning of each trial the positions and the orientations of the robots are randomly initialised, while the target areas are positioned systematically choosing a value in $D \in D_{set} = \{70, 90, 110, 130, 150\}$ cm (see Fig. 1). The performance F of the group during a trial is obtained evaluating how often and how quickly each robot in the group moves from one area to the other. For this purpose, each robot i cumulates a reward f every time it enters in a target area different from the one previously visited. This reward is computed according to the energy e_i saved in moving from one target area to the other.

$$f_i(t) = f_i(t-1) + \begin{cases} e_i(t) & \text{if robot enters a new target area} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $f_i(t)$ is the reward cumulated at time t, and $e_i(t)$ is the energy saved. Equation (1) states that when a robot enters in a target area different from the one previously visited, it stores the current energy load $e_i(t)$ as reward. At the same time, the energy level is reset to a quantity proportional to the distance between the target areas. Otherwise, the robot consumes its energy while moving, proportionally to its speed:

$$e_i(t) = \begin{cases} 1 + E_{ab} & \text{if robot enters a new target area} \\ e_i(t-1) - \delta_i(t) & \text{otherwise} \end{cases} \quad (2)$$

where E_{ab} is the energy that a robot would consume to move in a straight line between the two target areas, and $\delta_i(t)$ is the energy consumed in a single time-step, proportional to the wheels speed[1]. With an optimal behaviour (moving straight between the two areas), a robot would store a quantity $e_i(t) = 1$ each time it enters a new target area, independently from the distance between the two. The performance of the robot is computed at the end T of the trial, and is normalised according to the maximum performance that can be achieved by a robot behaving optimally:

$$\overline{f}_i = f_i(T)/f_{max}, \qquad f_{max} = v_{max} \cdot T_{eff}/D_{ab} \quad (3)$$

where $D_{ab} = D - d$ is the minimum distance that must be covered between two target areas. Finally, the fitness of the group F in a trial is computed as the average across the group:

$$F = \sum_{i=1}^{N} \overline{f_i} \quad (4)$$

The final fitness of the genotype is the average of F over 15 different trials.

3 Obtained Results and Behavioural Analysis

The evolutionary process has been replicated 10 times—hereafter, evolutionary runs—starting from randomly generated populations. At the end of each evolutionary run, we selected a single genotype to be analysed thoroughly. To do so, we evaluated the performance of the best genotype of the last 100 generations, and we selected the one showing the highest average over 500 trials as the representative of each evolutionary run. The results are shown in Fig. 2. A qualitative analysis of the evolved behaviours reveals that 6 evolutionary runs out of 10 result in a good collective exploration and navigation behaviour (runs number 3, 1, 6, 4, 7 and 0). Two runs (number 5 and 9) produced sub-optimal strategies,

[1] This is in [0.0, 0.0025], which means a robot wastes at most 1 unit of energy in 400 time-steps, if moving at maximum speed.

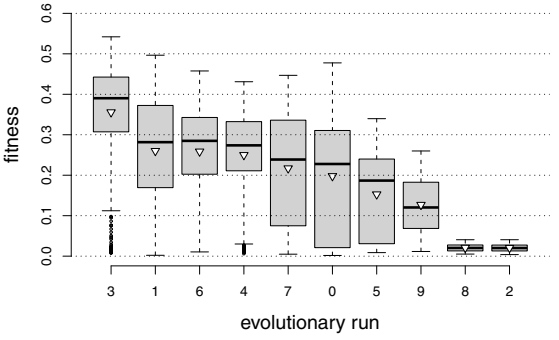


Fig. 2. Performance of the best evolved individual for each evolutionary run. Each boxplot corresponds to the performance obtained in 500 trials. Boxes represent the inter-quartile range of the data, while the horizontal lines inside the boxes mark the median values. The whiskers extend to the most extreme data points within 1.5 times the inter-quartile range from the box. Circles mark the outliers. The symbol $\bigtriangledown$ indicates the average performance.

and two others (number 8 and 2) resulted in unsatisfactory behaviours both at the individual and collective level. Given that the successful runs produced qualitatively similar behaviours, in the following we describe the one of the best genotype, corresponding to the evolutionary run number 3 (see Fig. 2).[2]

The sequence displayed in Fig. 3(a) shows how a typical trial unfolds in time[3]. Initially, robots move independently and explore the environment. In doing so, they signal their position and heading to other robots keeping the front blue LED switched on, while the red LED is used only in certain conditions (see below). The visual interactions mediated by these signals allow the group to converge to a coherent motion between the two target areas. Eventually, the robots form two rows moving from one target to the other in opposite directions (see the last frame in Fig. 3(a)). We refer to this structured spatio-temporal pattern formed by the robots as *dynamic chain*. The term *dynamic* well illustrates two interesting features of this structure. Firstly, each robot in the chain is not static, but moves continuously along it, swinging between the target areas as requested by the fitness function. Secondly, the chain connecting the two targets adapts its shape according to the current distance D between areas: it adapts the chain direction by choosing the shortest path between the two areas, and adapts the inter-robot distance to fit all robots in the chain (see Fig. 3(b)).

This collective behaviour is the result of simple rules followed by each individual robot and encoded in the neural controller. When a robot has no objects in its perceptual field, it moves counterclockwise in large circles, the front blue LED

[2] In this case the best genotype belongs to the 475^{th} generation.
[3] See http://laral.istc.cnr.it/esm/sperati-etal-ANTS2010/ for videos

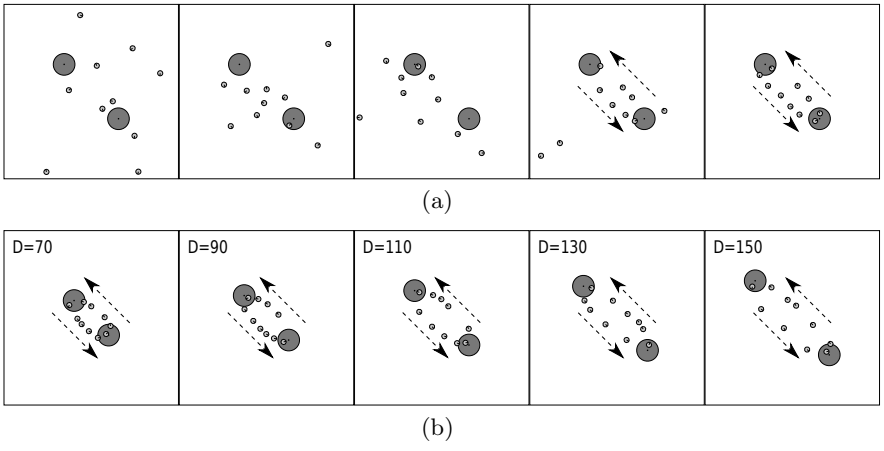


Fig. 3. (a) Temporal sequence recorded in a generic successful trial ($D = 110$ cm), showing the formation of the *dynamic chain*. (b) From left to right, each snapshot displays the final configuration achieved by the swarm, at the end of 5 different standard trials where the target areas are positioned according to the distance values in D_{set}.

always switched on. When a target area is in sight, a robot approaches it in a straight line and makes a u-turn when it reaches the grey circle. When two robots encounter, they avoid each other by always dodging to the right, exploiting the blue visual signal emitted by the robots. This constitutes the basic mechanism for the formation of the dynamic chain: in fact, a robot does not necessarily follow the robot in front moving in the same direction, but rather keeps on its left the robots coming in the opposite direction. It is clear that a minimal number of robots is necessary to support this behaviour, because a dynamic chain is stable as long as there are robots moving in opposite directions. In order to aggregate all robots in the chain formation, the red signal is exploited. In fact, the red signal mimics the colour of the target area, and in general induces an approaching behaviour. The red LED is switched on in two conditions. On the one hand, a robot flashes while moving towards a target area. In this case, the signal allows nearby robots to react by approaching the target area themselves, even though they do not directly perceive it. On the other hand, robots signal while avoiding each other. In this case, the function of the signal appears linked also to the enhancement of the stability of the dynamic chain. Although the identification of the exact roles played by the red and blue signals needs further analysis that we plan to carry on in future research, these observations clearly indicate that communication plays a crucial role for the formation of the dynamic chain and more generally for the ability of the robots to coordinate and cooperate.

This brief qualitative analysis suggests the following considerations. First of all, the formation of the chain is the outcome of a self-organising process that results solely from the robot-robot interactions. We observed that the dynamic chain forms rather abruptly out of a disordered group motion. We believe that

this may correspond to a phase transition that depends mainly on the density of robots between the two target areas. In other words, when enough robots are attracted in this area thanks to the visual communication, the chain forms. A second important remark concerns the function of the dynamic chain, which is exploited by the robots to maintain the right heading toward the target areas when they are not in sight. From this point of view, the group behaviour compensates for the limited sensory range of the robots, which collectively discover and preserve information concerning the direction of the two areas, thanks to the exploitation of the communication signals.

To better evaluate the performance of the group, we tested the collective behaviour systematically varying the distance between the target areas $D \in D_{set}$. The obtained results are presented in Fig. 4(a). Here the performance of the group is compared to the performance of a single individual evolved in a control experiment.[4] First of all, we notice that the group always outperforms the single individual. This confirms that a coordinated behaviour has been evolved, which goes beyond the capability of the individual robot. Looking at the performance of the group, it is possible to notice that the behaviour seems adapted mostly for an intermediate distance, in which it scores the highest average performance. With larger distances, the performance across different trials is more changing. This suggests that the group may be able to coordinate in some cases, and in others is not. We ascribe this variability to the limited duration of the trial, hypothesising that in some cases robots do not have enough time to coordinate and form a dynamic chain. To test this hypothesis, we performed an identical test, but now increasing the duration of the initial coordination period ($T_{add} = 18600, T_{eff} = 5400$ time-steps). The results plotted in Fig. 4(b) confirm that for all distances the group attains a good score, which is also very stable across different trials. Moreover, the results indicate that the group behaves better for large distances. In fact, with short distances, the dynamic chain is overcrowded and robots interfere with each other, therefore obtaining a lower performance. In this conditions, a smaller group behaves better (data not shown). Finally we note how, when $D = 150$, the performance of the individual continue to be very low, despite the extended time. This means that a robot alone is not capable to face all the D values, while the swarm is.

4 Generalisation Abilities

In the previous section, we have described the features of the evolved behaviour, and observed how the system always converges to a dynamic path formation if enough time is granted for coordination. In this section, we test the ability of the system to generalise to different conditions never met during the evolutionary optimisation. In particular, we want to understand whether robots are able to form

[4] We explicitly evolved a neural controller for a single robot confronted with the same task, with identical evolutionary conditions. We performed 30 evolutionary runs, each one lasting 1000 generations, and selected the best genotype with the same procedure described above.

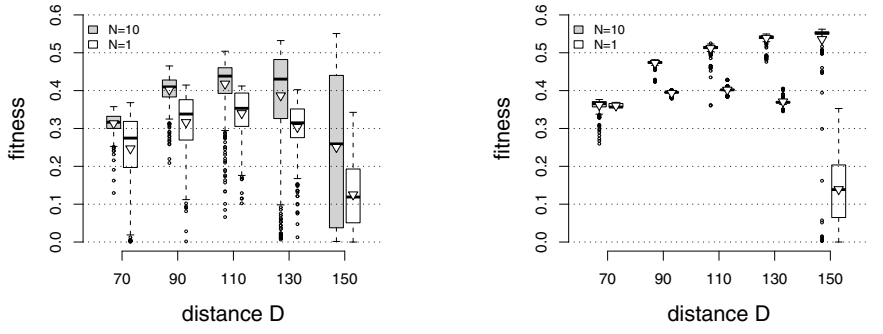


Fig. 4. (a) Performance test with varying distance $D \in D_{set}$. Performance is computed for groups of $N = 10$ robots, and for a single robot evolved for the same task in a control experiment ($N = 1$). (b) The same test performed with a longer initial period $T_{add} = 18600$, during which performance is not computed.

a path with larger distances and with larger groups. We test the performance of the group in 16 new conditions, obtained coupling 4 groups ($N \in \{20, 30, 40, 50\}$) with 4 distances ($D \in \{200, 250, 300, 350\}$ cm). These tests have been performed in a larger arena (fixed height $H = 350$ cm, variable width $W \in [350, 390]$ cm), and in longer trials ($T_{add} = 36600$). The quantitative results are shown in Fig. 5. We can immediately notice that, when the number of robots is sufficiently large, the swarm is successful also when the distance between the two target areas is much wider compared to the conditions experienced during the evolutionary process. With distance $D = 200$ cm, groups with 20 robots perform best, while larger groups are less efficient. Groups of 30 robots have a fairly good performance, which however presents a large variability. With distance $D = 250$ cm it is possible to notice a similar pattern. However, this time $N = 30$ is the optimal group size. Finally, for $D = 300$ cm and $D = 350$ cm, the size $N = 40$ performs best, with a larger variability in the latter case, in which also $N = 50$ presents a fairly good performance in many trials.

This analysis confirms our expectations: the larger the distance between the target areas, the larger the number of robots required to form a stable chain. In fact, as mentioned above, the dynamic chain is maintained as long as there are constantly robots moving in opposite directions uniformly distributed along the path, which implies larger groups for larger distances. The analysis also confirms that a minimum number of robots is necessary to form a path over a certain distance. Similarly, large groups suffer overcrowding when the distance D is too short, as there is no space available to distribute all the robots along the path. However, the dynamic chain can adapt to a wide range of distances. For instance, groups of 30 robots present good performance up to $D = 300$ cm, and only with larger distances the performance systematically drops.

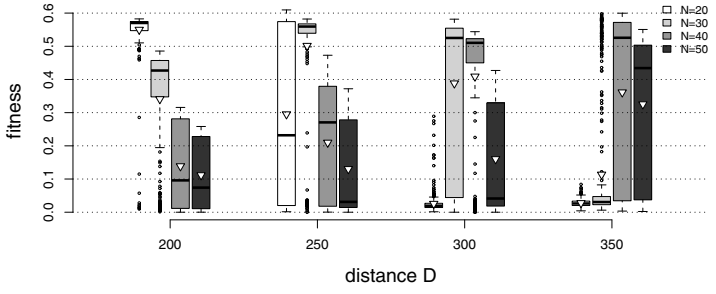


Fig. 5. Generalisation ability for groups of increasing size N and for increasing distance D. Each boxplot corresponds to the performance obtained in 500 trials ($T_{eff} = 5400$, $T_{add} = 36600$).

5 Discussion and Conclusions

In this paper we reported a series of experiments in which the behaviour of a swarm of robots has been evolved for the ability to navigate back and forth between two target areas which can only be perceived locally. The analysis of the obtained results indicates that the robots solve the problem by exploring the environment and by forming a dynamic chain constituted by two rows of robots moving from one target to the other in opposite directions. The dynamic chain emerges rather abruptly from the robot-robot interactions mediated by light signals, and afterwards adapts by converging toward a configuration that corresponds to the shortest path and that is characterised by a rather uniform distribution of distances between the robots. The analysis of the evolved behaviour demonstrates how it generalises to situations in which the distance between the two target areas is much wider compared to the conditions experienced during the evolutionary process provided that the number of robots forming the swarm is sufficiently large. Similarly to pheromone trails in ants, dynamic chains allow the swarm to efficiently navigate between the two target areas. Indeed, in both cases the stability of the structure is a result of a sustained flux of individuals that support it, in one case by pheromone laying, in the other by coloured signals. Another common feature is the ability to identify the shortest path between two locations. Even though we only performed tests in an obstacle-free arena, we observed that dynamic chains may initially form in curved paths (especially with large groups), which slowly straighten until the shortest route is taken. In future work, we plan to analyse in more detail the evolved behaviour, in order to better understand the properties of the dynamic chain and its relationship with similar behaviours observed in Nature.

In future work, we plan to test the evolved behaviour in hardware exploiting the foot-bot robotic platform developed within the European project *Swarmanoid* (grant IST-022888, see `http://www.swarmanoid.org`). Within this

project, we also aim at testing coordinated behaviours among groups of heterogeneous robots. In particular, we plan to exploit the eye-bot robotic platform to work as 'smart' target areas. The eye-bot is in fact an aerial robot with the ability to attach to the ceiling, from which it can monitor the environment and detect relevant areas. Foot-bots can perceive an eye-bot only when they are approximately underneath it, thanks to a camera pointing upward. As a consequence, eye-bots could be exploited by the foot-bots as target areas, and dynamic chains can be formed between them, creating for instance a delivery line from a target to a goal location, which is known by the eye-bots thanks to their privileged viewpoint. Moreover, eye-bots can move in the environment and exploit the robustness of the dynamic chain to purposely modify its length or its shape. Finally, we will investigate how to form more complex dynamic chains that can trace the shortest path between more than two eye-bots or target areas.

Acknowledgements. The authors wish to thank Onofrio Gigliotta, Tomassino Ferrauto and Gianluca Massera for the fruitful discussions and their help. This work was supported by the Swarmanoid project, funded by Future and Emerging Technologies programme (IST-FET), of the European Commission, under grant IST-022888.

References

1. Bailey, T., Durrant-Whyte, H.: Simultaneous localization and mapping: part II. IEEE Robotics & Automation Magazine 13(3), 108–117 (2006)
2. Detrain, C., Denebourg, J.: Collective decision and foraging patterns in ants and honeybees. Advances in Insect Physiology 35, 123–173 (2009)
3. Drogoul, A., Ferber, J.: From Tom Thumb to the Dockers: some experiments with foraging robots. In: From Animals to Animats 2, Second International Conference on Simulation of Adaptive Behavior (SAB-92), pp. 451–459. MIT Press, Cambridge (1993)
4. Filliat, D., Meyer, J.: Map-based navigation in mobile robots - I. A review of localization strategies. Journal of Cognitive Systems Research 4, 243–282 (2003)
5. Floreano, D., Husband, P., Nolfi, S.: Evolutionary robotics. In: Siciliano, B., Oussama, K. (eds.) Handbook of Robotics, pp. 1423–1451. Springer, Berlin (2008)
6. Gigliotta, O., Nolfi, S.: On the coupling between agent internal and agent/environmental dynamics: Development of spatial representations in evolving autonomous robots. Adaptive Behavior 16, 148–165 (2008)
7. Goss, A., Aron, S., Denebourg, J., Pasteels, J.: Self-organized shortcuts in the argentine ant. Naturwissenschaften 76(12), 579–581 (1989)
8. Gutiérrez, Á., Campo, A., Santos, F.C., Pinciroli, C., Dorigo, M.: Social odometry in populations of autonomous robots. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) ANTS 2008. LNCS, vol. 5217, pp. 371–378. Springer, Heidelberg (2008)
9. Hafner, V.V.: Cognitive maps in rats and robots. Adaptive Behavior 13, 87–96 (2005)
10. Lambrinos, D., Kobayashi, H., Pfeifer, R., Maris, M., Labhart, T., Wehner, R.: An autonomous agent navigating with a polarized light compass. Adaptive Behavior 6(1), 131–161 (1997)

11. Maynard-Smith, J., Harper, D.G.: Animal Signals. Oxford University Press, Oxford (2003)
12. Menzel, R., Greggers, U., Smith, A., Berger, S., Brandt, R., Brunke, S., Bundrock, G., Hülse, S., Plümpe, T., Schaupp, F., Schüttler, E., Stach, S., Stindt, J., Stollhoff, N., Watzl, S.: Honey bees navigate according to a map-like spatial memory. Proceedings of the National Academy of Sciences of the United States of America 102(8), 3040–3045 (2005)
13. Nolfi, S., Floreano, D.: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. MIT Press/Bradford Books, Cambridge (2000)
14. Nouyan, S., Campo, A., Dorigo, M.: Path formation in a robot swarm. Swarm Intelligence 2(1), 1–23 (2007)
15. O'Keefe, J., Nadel, L.: The Hippocampus as a Cognitive Map. Oxford University Press, Oxford (1978)
16. Ostergaard, E., Sukhatme, G., Matarić, M.: Emergent bucket brigading: a simple mechanisms for improving performance in multi-robot constrained-space foraging tasks. In: Proceedings of the Fifth International Conference on Autonomous Agents, pp. 2219–2223 (2001)
17. Payton, D., Daily, M., Estkowski, R., Howard, M., Lee, C.: Pheromone robotics. Autonomous Robots 11(3), 319–324 (2001)
18. Thrun, S.: Robotic mapping: a survey. In: Gerhard Lakemeyer, G., Nebel, B. (eds.) Exploring artificial intelligence in the new millennium, pp. 1–35. Morgan Kaufmann Publishers Inc., San Francisco (2003)
19. Trianni, V., Nolfi, S., Dorigo, M.: Evolution, self-organisation and swarm robotics. In: Blum, C., Merkle, D. (eds.) Swarm Intelligence. Introduction and Applications. Natural Computing Series, pp. 163–192. Springer, Berlin (2008)
20. Vickerstaff, R.J., Di Paolo, E.A.: Evolving neural models of path integration. Journal of Experimental Biology 208, 3349–3366 (2005)
21. Wehner, R.: Desert ant navigation: how miniature brains solve complex tasks. Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology 189(8), 579–588 (2003)
22. Werger, B., Matarić, M.: Robotic food chains: Externalization of state and program for minimal-agent foraging. In: From Animals to Animats 4, Fourth International Conference on Simulation of Adaptive Behavior (SAB 1996), pp. 625–634. MIT Press, Cambridge (1996)
23. Zeil, J., Boeddeker, N., Stürzl, W.: Visual homing in insects and robots. In: Floreano, D., Zufferey, J.C., Srinivasan, M., Ellington, C. (eds.) Flying Insects and Robots, pp. 87–100. Springer, Berlin (2009)

Extensions to the Ant-Miner Classification Rule Discovery Algorithm

Khalid M. Salama and Ashraf M. Abdelbar

Computer Science & Engineering Department
American University in Cairo, Egypt
{khalid.magdy,abdelbar}@aucegypt.edu

Abstract. Ant-Miner is an ant-based algorithm for the discovery of classification rules. This paper proposes four extensions to Ant-Miner: 1) we allow the use of a logical negation operator in the antecedents of constructed rules; 2) we use stubborn ants, an ACO-variation in which an ant is allowed to take into consideration its own personal past history; 3) we use multiple types of pheromone, one for each permitted rule class, i.e. an ant would first select the rule class and then deposit the corresponding type of pheromone; 4) we allow each ant to have its own value of the α and β parameters, which in a sense means that each ant has its own individual personality. Empirical results show improvements in the algorithm's performance in terms of the simplicity of the generated rule set, the number of trials, and the predictive accuracy.

Keywords: Ant Colony Optimization (ACO), Data Mining, Classification, Stubborn Ants, Ants with Personality.

1 Introduction

Classification is a data mining task in which the aim is to discover, from labeled cases, a model that can be used to predict the class of unlabeled cases [4]. Ant-Miner is an ACO algorithm, proposed by Parpinelli et al. [9], that discovers classification rules of the form:

IF *<Term-1>* ***AND*** *<Term-2>* ***AND*** *...<Term-n>* ***THEN*** *<Class>*,

where each term is of the form *<attribute = value>*, and the consequent of a rule is the predicted class. In this paper, we propose a number of extensions to the Ant-Miner algorithm, and then empirically evaluate each of these, both individually and in combination with the others.

In section 2, we present a brief description of the original Ant-Miner algorithm, followed by a brief review of related work in Section 3. We then present each of our proposed extensions in Sections 4 through 7. Section 4 describes the use of a logical negation operator in the construction of rule antecedents. Section 5 proposes the use of stubborn ants, where an ant is influenced by its own rule construction history. In section 6 we introduce the multi-pheromone ant system, in which an ant selects the rule class first and then drops different

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 167–178, 2010.

pheromone types for each selected class. Finally, in Section 7, we explore the idea of using different values for α and β for each ant. Sections 8 and 9 discuss our experimental methodology and results, respectively, and some final remarks are presented in Section 10.

2 Ant-Miner Algorithm

The proposed modifications presented in this paper are based on the original Ant-Miner algorithm introduced in [9]. The following is a brief description of the algorithm; for a more detailed discussion, the reader is referred to [9]. Ant-Miner discovers an ordered list of classification rules. As an ACO-based algorithm, the decision components in the construction graph of Ant-Miner are the available attribute values, by which a rule's antecedent terms can be constructed. The algorithm consists of two nested loops: the outer loop where a single rule in each iteration is added to the discovered rule list, and the inner loop where an ant in each iteration constructs a rule as follows. Each ant in the colony attempts to construct a rule's antecedents by selecting terms probabilistically according to a heuristic function involving information gain [10] and the pheromone amount for this term, until all the attributes have been used, or until adding any other term to the set of antecedents would make the rule coverage less than `min_cases_per_rule`. The rule consequent is then chosen by determining the class value with maximum occurrence in the cases matching the rule antecedents. A pruning process is carried out on the rule to increase the rule quality. Then, the ant updates the pheromone level by depositing pheromone on the selected terms in proportion to the quality of the rule. This is done in order to increase the probability that the following ants will select the terms involved in the rule. The pheromone values for all terms are normalized to simulate evaporation. After several iterations of the inner loop, the best rule constructed is added to the list of discovered rules, and the training cases matched by that rule are removed from the training set. This course of action is considered an iteration of the outer loop and is repeated until the number of training examples remaining in the training set becomes less than or equal to the value determined by the `max_uncovered_cases` parameter. Both `min_cases_per_rule` and `max_uncovered_cases` are user-specified thresholds.

2.1 Pheromone Initialization and Update

At the beginning of each outer loop, the pheromone is initialized for each term with the same value given by the function:

$$\tau_{ij}\,(t=0) = \frac{1}{\sum_{r=1}^{a} b_r} \tag{1}$$

where a is the total number of attributes, i is the index of an attribute, j is the index of a value in the domain of attribute i, and b_r is the number of values in the domain of attribute r.

After an ant constructs a rule, the rule quality is evaluated and the pheromone amount is increased for the terms belonging to the rule according to its quality. This is calculated as follows:

$$Q = \frac{TP}{TP + FN} \cdot \frac{TN}{TN + FP} \tag{2}$$

where TP (true positives) is the number of cases covered by the rule and have the class predicted by the rule, FP (false positives) is the number of cases covered by the rule and have a class different from the class predicted by the rule, FN (false negatives) is the number of cases that are not covered by the rule but have the class predicted by the rule, TN (true negatives) is the number of cases that are not covered by the rule and do not have the class predicted by the rule.

2.2 Term Selection

The term is selected probabilistically according to two components. The first is η_{ij}, which is the value of a problem-dependent heuristic function – information gain [10] is used in Ant-Miner – for $term_{ij}$. The second is τ_{ij}, which is the amount of pheromone on $term_{ij}$. The higher the value of η_{ij} is, the better for classification the $term_{ij}$ is, thus leading to a higher probability of being selected. The same applies for τ_{ij}. The following is the term selection formula:

$$P_{ij} = \frac{\eta_{ij} \cdot \tau_{ij}(t)}{\sum_{r=1}^{a} \sum_{s=1}^{b_r} (\eta_{rs} \cdot \tau_{rs}(t))} \tag{3}$$

where P_{ij} is the probability of selecting $term_{ij}$, a is the total number of attributes, and b_r is the number of values in the domain of the r^{th} attribute.

3 Related Work

A. Chan and A. Freitas [2] have proposed a new rule pruning procedure for Ant-Miner that led to the discovery of simpler (shorter) rules and improved the computational time in datasets with a large number of attributes. AntMiner2 and AntMiner3 have been introduced in [5] and [6] respectively by B. Liu , H. A. Abbass, and B. McKay which employ a density-based heuristic function for calculating the heuristic value for a term, as well as presenting a new state transition approach respectively. A pseudorandom proportional transition rule was used by Z. Wang in [13]. J. Smaldon, and A. Freitas [11] introduced the idea of selecting rule consequent class before rule construction - this idea is the inspiration of multi-pheromone ant system modification described in section 7 - and producing an unordered rule set. D. Martens [7] has introduced a new ACO-based classification algorithm, named AntMiner+. It makes a distinction between nominal and ordinal attributes. Instead of creating a pair (attribute = value) for each value of an ordinal attribute, AntMiner+ creates two types of bounds that represent the interval of values to be chosen by the ants. Moreover,

it employs different pheromone initialization and update procedures based on the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System [3]. In addition, edges in the construction graph are considered the decision components. The idea of selecting rule class before constructing rule antecedent is also used in his paper. F. Otero, A. Freitas, and C. G. Johnson [8] introduced a version of Ant-Miner that copes with continuous attributes named cAnt-Miner.

4 Using Logical Negation Operator in Rule Antecedents

In the original and various versions of Ant-Miner, the construction graph consists of nodes representing attribute values of the problem domain. The set of nodes (N) in the construction graph is:

$$N = \bigcup_{i=1}^{n} v_{ij} \ , \quad j \ \in \{1, 2, \ldots, l\}$$

where i is the i^{th} attribute, n is the number of attributes and v_{ij} is the j^{th} permitted value of the i^{th} attribute. Thus the constructed rule antecedents will be in the form of:

$$\boldsymbol{IF} < A_i \ = V_{ij} > \boldsymbol{AND} < A_k \ = V_{kl} > \boldsymbol{AND} \ \ldots$$

To allow using the logical negation operators in the antecedents of constructed rules, the values and their negation per attribute will be added to the construction graph. The set of nodes (N) in the construction graph will be:

$$N = \bigcup_{i=1}^{n} v_{ij} \ \cup \ \bigcup_{i=1}^{n} \overline{v_{ij}}, \quad j \ \in \{1, 2, \ldots, l\}$$

Thus, the available decision components in the construction graph allow constructing rule antecedents in the form of:

$$\boldsymbol{IF} < A_i \ = V_{ij} > \boldsymbol{AND} < A_k \ \boldsymbol{NOT} \ = V_{kl} > \boldsymbol{AND} \ldots$$

Negation values are added for the attribute that has more than two values in its domain. Pheromone is updated regularly on these terms and a heuristic value is calculated for the negation attribute values in the same way as it is calculated for regular attribute values. An example of a generated rule using logical negation operator is: "IF <price = low> AND <condition NOT = bad> THEN <Class=Buy>".

Although using negative attributes doubles the size of the construction graph, it enables the construction of rules that have greater coverage of the training cases. Consequently, a lower number of rules is produced which improves the comprehensibility of the output. Moreover, a reduced number of iterations are needed to reach the threshold of the number of cases to be covered. Results also show that it has a better performance in terms of accuracy in addition to the reduced number of iterations and the simpler (smaller) rule set.

5 Using Stubborn Ants

Stubborn ants were introduced in 2008 in [1]. The idea is to promote search diversity by having each ant be influenced by its own history of constructing solutions in addition to the pheromone trails left by other ants. Basically, each ant does several trials in the execution of the algorithm. Each ant_t memorizes the best solution R_t^+ that it has constructed during its own trials. If $term_{ij}$ belongs to the antecedents of rule R_t^+, then $term_{ij}$ will have an amplified probability of being selected by ant_t, with the degree of amplification depending on the quality of the solution R_t^+. The probability that a term will be added to the current rule is given by the following formula:

$$P_{ij}(t) = \frac{V_{ij}}{\sum_{r=1}^{a} \sum_{s=1}^{b_r} (V_{rs})} \tag{4}$$

where $V_{ij} = (\eta_{ij} \cdot \tau_{ij}(t)) + (\eta_{ij} \cdot \tau_{ij}(t)) \cdot Q(R_t^+)$ if $term_{ij}$ belongs to ant_t history's best rule R_t^+, otherwise $V_{ij} = \eta_{ij} \cdot \tau_{ij}(t)$, and $Q(R_t^+)$ represents the quality of the current ant's best history rule R_t^+.

Stubborn ants add individuality to each ant, which promotes exploration and diversity. Results (see Section 9) show an increase in rule accuracy when using stubborn ants as well as a decrease in the average number of trials needed to construct a rule. Note that the size of the colony affects the behavior of the stubborn ants; as the number of the ants decreases, the stubbornness effect is more applied, given that the total number of trials per iteration is fixed.

6 Multi-Pheromone Ant System

In the original Ant-Miner, the consequent of a rule is chosen after its antecedents are selected by determining the class value with maximum occurrence in the cases matching the rule premises. The idea of selecting the rule consequent prior to rule construction was introduced in different flavors. A. Frietas in [11] introduced an algorithm that tries to construct rules for each class independently: an extra For-Each (class value) loop is added as an outer loop for the original algorithm. The consequent of the rule is known by the ant during rule construction and does not change. An ant tries to choose terms that will produce the rule predicting the class value in the current iteration of the For-Each loop with an optimum level of accuracy. This approach generates better rules in comparison with the original Ant-Miner where a term is chosen for a rule in order to decrease entropy in the class distribution of cases matching the rule under construction. However, the entire execution (with the complete training set) is repeated separately for each class value until the number of positive examples (belonging to the current class) remaining in the dataset that have not been covered by the discovered rules is less than or equal to `max_uncovered_cases`. Moreover, the number of the generated rules by this version is increased. For a more detailed description of the algorithm, refer to [11].

D. Martens introduced the same idea in Ant-Miner+ [7]. An extra vertex group is added at the start in the construction graph containing class values to allow the selection of class first. This is similar to considering the class as another variable. Rules with different classes can be constructed in the same iteration. Different heuristic values are applied according to the selected class in order to chose the term that is relevant to the prediction of the selected class. However, the pheromone is shared by all ants constructing rules with different consequents. In other words, any ant is influenced by the pheromone dropped by any other ant constructing similar or different labeled rules. This can negatively affect the quality of the constructed rules, as the terms that lead to constructing a good rule with class C_x as a consequent do not necessarily lead to constructing a good rule with C_y as a consequent for a classification rule.

Unlike the version of Ant-Miner in [11], our proposed multi-pheromone Ant-Miner system executes the course of operations only once during the entire training process. Ants in the multi-pheromone system can construct rules with different consequent classes in the same iteration simultaneously. Nonetheless, the ant is only influenced by the ants that have constructed rules with the same consequent, using a multiple types of pheromone system.

First, an ant probabilistically selects the rule consequent prior to antecedents based on pheromone information as described below. Then, it tries to chose terms that are relevant to predicting this class. The rule is then evaluated and the pheromone is updated. But, unlike the version of Ant-Miner in [7], the ant drops different kinds of pheromone as many as the permitted classes. The next ant is only influenced by the amount of the pheromone deposited for the class for which it is trying to construct a rule. In this case, pheromone is not shared amongst ants constructing rules for different classes. This allows choosing terms that are only relevant to the selected class.

The idea of multi-pheromone Ant-Miner is that each class has a different pheromone to be deposited on the terms in the construction graph. In essence, we are replacing the traditional two-dimensional pheromone structure (attribute, value) by a new three-dimensional pheromone structure (attribute, value, class). During rule construction, the rule class is already set and an ant is only influenced by the amount of pheromone in the pheromone array element dedicated to its rule class. Similarly in pheromone update, an ant deposits pheromone in the array element dedicated to the current rule class in each node belonging to the trial. Class values are also represented in nodes in the construction graph, and pheromone can be deposited on them. This pheromone affects the probability of selecting the rule class for subsequent ants. The pheromone is initialized in the node of class values as follows:

$$\tau_c = \frac{freq(c)}{|TrainingSet|} \tag{5}$$

where $freq(c)$ is the number of instances labeled with class c, and $|TrainingSet|$ is the size of the training set. In pheromone update, the pheromone level increases in the node of the constructed rule class according to the quality of the rule, as follows:

$$\tau_{c_r}(t) = \tau_{c_r}(t-1) + \tau_{c_r}(t-1) \cdot Q_r \tag{6}$$

where c_r is the class of the current constructed rule, and Q_r is quality of the rule. The problem dependent heuristic function chosen is the Laplace-corrected confidence for each term as in [11], given by:

$$\eta_{ij,k} = \frac{|term_{ij},\ k| + 1}{|term_{ij}| + \texttt{no\_of\_classes}} \tag{7}$$

where $\eta_{ij,k}$ is the heuristic for $term_{ij}$ given that class k is selected, $|term_{ij}, \mathrm{k}|$ is the number of training cases having $term_{ij}$ and the current selected class k, $|term_{ij}|$ is the number of training cases having $term_{ij}$ and `no_of_classes` is the number of values in the class attribute's domain. The probability of selecting $term_{ij}$ given that class k is chosen is calculated as follows:

$$P_{ij,k} = \frac{\eta_{ij,k} \cdot \tau_{ijk}(t)}{\sum_{r=1}^{a} \sum_{s=1}^{b_r} (\eta_{rs,k} \cdot \tau_{rs,k}(t))} \tag{8}$$

The rule generated via multi-pheromone system is evaluated, to update the pheromone levels, by a function that balances between the support and the confidence of the rule, as follows:

$$Q(R_t) = Confidence(R_t) + Support(R_t) \tag{9}$$

where $Confidence(R_t) = \frac{TP}{|Matches|}$ represents the ratio of the number of cases that match rule R_t's premises and are labeled by its class to the total number of cases that match R_t's premises, and $Support(R_t) = \frac{TP}{|TrainingSet|}$ represents the ratio of the number of cases that match R_t's premises and are labeled by its class to the total number of cases in the training set. This function finds a middle ground between the coverage of the rule and its classification accuracy.

After the best iteration rule is selected, all cases covered by this rule are removed from the training set and the pheromone is initialized but only in the pheromone array element dedicated for the class of this rule. Leaving the pheromone in the array element of other classes tends not to waste the wisdom that has been collected by the ants in the previous trials for the rest of the classes, leading to faster convergence in the next iterations. Multi-pheromone Ant-Miner system tends to generate better rules (see Section 9 for results) in terms of accuracy with a smaller number of iterations and generates a smaller (simpler) rule set.

7 Ants with Personality

Ants with personality were proposed in the future work section of [1]. In typical ACO systems, the probabilistic transition function is calculated as follows:

$$P_i = \frac{\eta_i^{\alpha} \cdot \tau_i^{\beta}(t)}{\sum_{r=1}^{a} \left(\eta_r^{\alpha} \cdot \tau_r^{\beta}(t)\right)} \tag{10}$$

The exponents α and β are used to adjust the relative emphases of the pheromone and heuristic information terms, respectively. In the original ant-miner, α equals β equals to 1.0, for all ants.

In ant with personality [1], the idea is to increase search diversity by giving each ant its own values of the α and β parameters, different from those of the rest of the colony. In our experimental results, we use values of α and β drawn from a random number generator using a Gaussian distribution with a mean of 2 and a standard deviation (σ) that ranges from 0 to 1. Note that a higher standard deviation value would introduce a higher range of diversity between ant behavior in selecting a term. However, this could also increase the number of trials needed in each iteration to converge on a rule.

8 Experimental Methodology

The performance of Ant-Miner with the proposed modifications was evaluated using four public-domain datasets from the UCI (University of California at Irvine) dataset repository [12].

The main characteristics of the datasets are shown in Table 1. The new version of Ant-Miner does not deal directly with continuous attributes, therefore, the chosen datasets include only categorical attributes in order to avoid the interference of the quality of the discretization method on the experiment.

Table 1. Description of Datasets Used in Experimental Results

Dataset	# of cases	# of attributes	# of classes
Car Evaluation	1,728	6	4
Tic-Tac-Toe	958	9	2
Mushrooms	8,124	22	2
Nursery	12,960	8	5

Ten-fold cross validation was used to split the dataset into a training set and testing set with ratio of 90% and 10% respectively. Each pair of training and testing data was used for experimenting with each combination of modifications (original, using negative attributes, using stubborn ants and multi-pheromone) and the average was taken. The experiment ran 10 times per each pair and the average of averages was taken. Thus, the total number of runs for each dataset is 100 (10 pairs, each tested 10 times). the number of rules generated (which represents the comprehensibility of the output), the average number of trials per iteration (number of ant trials needed to converge) and the accuracy of the generated rules were recorded to evaluate the quality of the experiment.

For ants with personality, the algorithm has been executed on the four datasets with different values for the standard deviation parameter (σ). Each value of standard deviation is tried 10 times for each training/testing pair taken from each dataset.

The source code for our extended version of Ant-Miner, including all four extensions presented in this paper is available at the following address: `http://www.aucegypt.edu/faculty/abdelbar/ant-miner-extended.zip`

Table 2 shows the algorithm parameter settings used in the experiments for testing the original version as well as the various proposed extensions of the Ant-Miner.

Table 2. Algorithm Parameters Used in Experiments

Parameter	Value
Number of ants	5
`max_uncovered_cases`	5%
Number of trials per ant	300
Number of trials to test converge (`no_rules_converg`)	10
Number of global iterations	30

9 Experimental Results

The following are the results produced by applying the new modifications on the chosen datasets. Results for each dataset are presented in a separate table. Each table contains experimental results for applying each modification individually and in combination with others. The results for ants with personality are presented in a separate table which contains the results for each dataset using different values of standard deviation σ.

As shown in Table 3, using logical negation reduced the average number of rules generated by the algorithm. Stubborn ants improved the average accuracy of the generated rules and reduced the average number of trials per iteration. Multi-pheromone system improved the average accuracy with most the scenarios compared to the original version. Using Multi-pheromone with stubborn ants and logical negation produced the best average accuracy with a reduced number of rules and a smaller number of trials per iteration.

In the Tic-Tac-Toe dataset (Table 4), the class attribute has two values. Multi-pheromone did not improve the accuracy of the generated rules. However, it produced a smaller rule set. Using logical negation reduced the average number of

Table 3. Experimental Results for the Car Evaluation Dataset

	Original			Multi-Pheromone		
	# Rules	Trials/Iter	Accuracy	# Rules	Trials/Iter	Accuracy
None	8.9	87	75.7 %	6.1	154	77.3 %
Negation	5.9	105	78.0 %	**3.7**	169	76.9 %
Stubborn Ants	8.6	68	77.6 %	5.6	91	78.7 %
Neg. & Stub.	6.2	84	79.1 %	4.8	114	**80.3%**

Table 4. Experimental Results for the Tic-Tac-Toe Dataset

	Original			Multi-Pheromone		
	# Rules	Trials/Iter	Accuracy	# Rules	Trials/Iter	Accuracy
None	6.6	89	70.1 %	5.8	148	69.9 %
Negation	5.3	120	70.6 %	**3.1**	109	70.8 %
Stubborn Ants	6.9	59	71.5 %	5.9	83	70.3 %
Neg. & Stub.	4.9	97	**72.7%**	3.2	99	71.8 %

generated rules. Stubborn ants enhanced the average accuracy of the rules. Using logical negation with stubborn ants in the original version produced the best average accuracy while using multi-pheromone with logical negation produced the least number of rules.

The Mushrooms dataset (Table 5) has a two-valued class attribute, as in Tic-Tac-Toe. However, multi-pheromone system produced better results in terms of average accuracy. Stubborn ants performed well in enhancing the average accuracy of the generated rules. Using logical negation produced the least number of rules with a low number of trials, but the average accuracy of the rules declined. Multi-pheromone with stubborn ants produced the best average accuracy with an appropriate number of generated rules.

Experiments on the Nursery dataset (Table 6) have shown similar results to the Mushrooms dataset. Using logical negation reduced the number of generated rules, but came with a negative effect on the accuracy. Stubborn ants improved

Table 5. Experimental Results for the Mushrooms Dataset

	Original			Multi-Pheromone		
	# Rules	Trials/Iter	Accuracy	# Rules	Trials/Iter	Accuracy
None	6.0	41	91.0 %	4.3	132	91.3 %
Negation	4.4	59	88.5 %	3.8	277	89.6 %
Stubborn Ants	6.5	38	92.6 %	4.7	87	**93.3%**
Neg. & Stub.	4.9	39	90.2 %	**3.5**	139	91.7 %

Table 6. Experimental Results for the Nursery Dataset

	Original			Multi-Pheromone		
	# Rules	Trials/Iter	Accuracy	# Rules	Trials/Iter	Accuracy
None	9.0	115	78.4 %	7.3	215	79.1 %
Negation	6.2	110	76.4 %	**4.2**	253	75.1 %
Stubborn Ants	9.1	95	79.2 %	6.6	147	**80.5%**
Neg. & Stub.	5.7	90	76.0 %	5.1	237	77.6 %

Table 7. Experimental Results for Ants with Personality

	Car Evaluation		Nursery		Tic-Tac-Toe		Mushrooms	
	$\sigma = 0.5$	$\sigma = 1.0$	$\sigma = 0.5$	$\sigma = 1.0$	$\sigma = 0.5$	$\sigma = 1.0$	$\sigma = 0.5$	$\sigma = 1.0$
Avg. Accuracy	76.2%	77.6%	80.3%	81.0 %	71.6%	72.8 %	92.8%	91.7%
Avg. Trials/Iter.	156	302	230	879	198	634	167	538
Avg. # Rules	9	8.8	9.2	9.1	6.7	6.7	6	6

the average accuracy of the generated rules, especially when used with multi-pheromone system, as this combination produced the best average accuracy.

As for ants with personality, Table 7 indicates that the average number of rules did not change significantly when using different values of standard deviation for all datasets. However, the average accuracy and the number of trials have different results for each value of standard deviation. Setting $\sigma = 0.5$ produced better average accuracy compared to the original version of Ant-Miner but with higher trials per iteration. A standard deviation of 1.0 produced an even better average accuracy, but the number of trials per iteration also increased.

In summary, experimental results indicate that using logical negation tends to produce a lower number of rules. However, since the number of nodes in the construction graph increases, the number of trials per iteration increases. Using logical negation does not sacrifice the accuracy of the generated rules. On the other hand, using stubborn ants enhances the classification accuracy of the rules. Multi-pheromone increases rule quality in terms of accuracy. Furthermore, it produces a smaller rule set because of the evaluation function that balances between a rule's classification accuracy and its coverage. As for ants with personality, using $\sigma = 1.0$ produces better results in terms of accuracy than using $\sigma = 0.5$. Nonetheless, the algorithm needs less trials using $\sigma = 0.5$. Note that a standard deviation of 0.5 produces better results in terms of generated rules accuracy compared to the original version of Ant-Miner.

10 Concluding Remarks

This paper has proposed four extensions to the Ant-Miner classification rule discovery algorithm. Experimental results on four popular datasets indicate that these extensions are promising and worthy of further exploration.

In the future, we would like to explore using a weight coefficient for stubbornness when using stubborn ants, which may start at a small value and increase gradually over time. When using ants with personality, we would like to explore gradually decreasing over time the value of the standard deviation of the Gaussian distribution function used to generate the individual α's and β's. Another research direction is to enhance the pheromone update procedure by rewarding a rule whose quality is higher than a certain threshold by depositing more pheromone and penalizing a low-quality rule by removing pheromone from its terms in the construction graph.

References

1. Abdelbar, A.M.: Stubborn ants. In: Proceedings IEEE Swarm Intelligence Symposium, pp. 1–5 (2008)
2. Chan, A., Freitas, A.: A new classification-rule pruning procedure for an ant colony algorithm. In: Talbi, E.-G., Liardet, P., Collet, P., Lutton, E., Schoenauer, M. (eds.) EA 2005. LNCS, vol. 3871, pp. 25–36. Springer, Heidelberg (2006)
3. Dorigo, M., Colorni, A., Maniezzo, V.: The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B 26, 29–41 (1996)
4. Jaiwei, H., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2006)
5. Liu, B., Abbass, H.A., McKay, B.: Density-based heuristic for rule discovery with ant-miner. In: Proc. 6th Australasia-Japan Joint Workshop on Intell. Evol. Syst., pp. 180–184 (2002)
6. Liu, B., Abbass, H.A., McKay, B.: Classification rule discovery with ant colony optimization. In: Proc. IEEE/WIC Int. Conf. Intell. Agent Technol., pp. 83–88 (2003)
7. Martens, D., Backer, M.D., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B.: Classification with ant colony optimization. IEEE Transactions on Evolutionary Computation 11, 651–665 (2007)
8. Otero, F., Freitas, A., Johnson, C.G.: cAnt-Miner: An ant colony classification algorithm to cope with continuous attributes. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) ANTS 2008. LNCS, vol. 5217, pp. 48–59. Springer, Heidelberg (2008)
9. Parpinelli, R.S., Lopes, H.S., Freitas, A.: Data mining with an ant colony optimization algorithm. IEEE Transactions on Evolutionary Computation 6, 321–332 (2002)
10. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
11. Smaldon, J., Freitas, A.: A new version of the Ant-Miner algorithm discovering unordered rule sets. In: Proceedings Genetic and Evolutionary Computation Conference (GECCO), pp. 43–50 (2006)
12. UCI Repository of Machine Learning Databases, `http://www.ics.uci.edu/~mlearn/MLRepository.html` (Retrieved July 2009)
13. Wang, Z., Feng, B.: Classification Rule Mining with an Improved Ant Colony Algorithm. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 357–367. Springer, Heidelberg (2004)

Functional Blueprints: An Approach to Modularity in Grown Systems

Jacob Beal

BBN Technologies, Cambridge, MA, USA
jakebeal@bbn.com

Abstract. The engineering of grown systems poses fundamentally different system integration challenges than ordinary engineering of static designs. On the one hand, a grown system must be capable of surviving not only in its final form, but at every intermediate stage, despite the fact that its subsystems may grow unevenly or be subject to different scaling laws. On the other hand, the ability to grow offers much greater potential for adaptation, either to changes in the environment or to internal stresses developed as the system grows. I observe that the ability of subsystems to tolerate stress can be used to transform incremental adaptation into the dynamic discovery of viable growth trajectories for the system as a whole. Using this observation, I propose an engineering approach based on *functional blueprints*, under which a system is specified in terms of desired performance and means of incrementally correcting deficiencies. I demonstrate this approach by applying it to integrate simplified models of tissue growth and vascularization, then further demonstrate how the composed system may itself be modulated for use as a component in a more complex design.

1 Introduction

One of the most remarkable facts about animals is that they are not generally injured by their own growth. An animal is composed of many tightly integrated systems, all interlocking in multiple ways. For example, bones fit together in joints that permit a useful range of motion, muscles attach to the bones in a pattern that allows them to work together effectively to move the body, the circulatory system delivers oxygen and nutrients to every portion of the bones and muscles via an intricate network of vessels, and their waste products are carried away for removal by the kidneys. As the animal grows, from an embryo to a mature adult, all of these systems are constantly adapting in order to remain integrated and fully functional.

This is not generally the case for our current engineered systems. Many artifacts, such as cars and airplanes, have no real capacity for growth at all. In engineered systems that do grow, the growth is often accompanied by significant degradation of function as the existing balance of systems is disrupted and painstakingly reintegrated. Adding an extension to a house means months of dust, being unable to use existing rooms, and electrical and plumbing disruptions. Expanding the road networks of a growing city requires years of detours

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 179–190, 2010.

and traffic disruptions, not to mention economic disruption for businesses nearby the construction. Upgrading the software of a computer often requires a reboot and leaves a trail of incompatibilities and ongoing headaches. Beyond the obvious differences in mechanical and material properties, we simply do not know how to describe our designs in a way that allows for disruption-free growth. We may thus be led to consider languages for adaptable design, both to better understand animal development and also to improve engineered systems. This is particularly pressing given the rapid progress occurring in synthetic biology (e.g. engineered pattern formation [2] and standardized DNA assembly protocols [13]), where the systematic engineering of DNA programs promises to soon allow us to created engineered objects that are literally grown from living cells.

One particularly elegant example of growth and adaptivity in biological systems is the vascular system [4]. Under normal conditions, sufficient oxygen diffuses through the walls of capillaries into the surrounding tissue. When cells are not receiving enough oxygen, however, they become stressed and emit a chemical signal that causes nearby capillaries to leak. The vascular system also has an elegant program for regulating its capacity. When a capillary leaks often, a new capillary begins to grow out of the leaky area, increasing the available blood supply to the oxygen-starved region. Blood vessels are elastic, and when they are frequently stretched, the cells divide, increasing the capacity of the vessel; likewise, when frequently contracted, cells die and shrink the vessel. Thus, the vascular system incrementally grows and shrinks to match the demand of the tissues it serves, branching into under-served regions and adjusting the size of vessels to match the flow through them.

In this paper, I propose an engineering approach of *functional blueprints* inspired by this and other similar adaptive biological systems. If each system is capable of operating under minor stress and of incrementally adjusting to decrease stress, then feedback between components should allow all the subsystems comprising a natural or engineered system to maintain a tight integration as the system grows, even if the relationship and relative sizes of subsystems are changing. Functional blueprints attempt to capture this by specifying a system in terms of desired performance and means of incrementally correcting deficiencies.

In the remainder of those paper, I first discuss how stress tolerance can enable integrated growth, then formalize this idea with a definition for a functional blueprint. I next demonstrate the functional blueprint approach by applying it to integrate simplified models of cell density maintenance and vascularization produce synchronized tissue growth, then finally show how the composed system may itself be modulated for use as a component in a more complex design.

1.1 Related Work

Morphogenesis in natural systems has been a subject of intensive study. In recent years, deciphering of genetic mechanisms controlling development, such as how the *hox* gene complex produces the overall body plan of animals, has lead to a synthesis of evolution and development (EvoDevo) [5], and theories of how the adaptivity of organisms to body plan variations may facilitate evolution [8].

Inspiration from natural systems has led to investigation of how growable patterns might be programmed, generally focusing on the establishment of shape, with less attention to integration of function. Doursat, for example, has developed a *hox*-gene based network model for artificial evolution of animal-like systems [7]. Similarly, the development of structure in growing plants has long been modeled at a high level by term-rewriting systems [12], which the MGS language extends into a general model of structure development through topological rewriting [14]. Other notable approaches include Coore's Growing Point Language [6], which uses a botanical metaphor to create topological structure and Nagpal's Origami Shape Language [11], which creates geometric forms through folding. Most similar to this work is Werfel's work on distributed construction, which has been extended to use functional constraints to generate adaptive structure in response to environmental stimuli [15].

The problems of integration addressed in this paper are also related to control theory. Standard control theory, however, has difficulty addressing systems with large numbers of non-linearly interacting parts, which are typical of growing systems. A notable exception may be viability theory [1], a branch of mathematical theory which is intended to address such concerns.

2 Stress Tolerance Enables Integrated Growth

The basic insight enabling this new approach is as follows: a stress-tolerant system can exploit its tolerance to navigate dynamically through the space of viable designs. This is rather foreign to the typical engineering approach to failure tolerance. Usually, an engineer designing a system treats its ability to tolerate failures like guard rails on a highway: important for safety, changing terrible outcomes into merely bad, but never touched under normal circumstances. Alternately, though, we can treat the system's robustness as a guide, the way that a blind person, might use a guard rail to follow the twists and turns of the road.

Under this alternate view, stress within the system becomes the coordinating signal by which independently developing subsystems are integrated. When the system is far from the edge of its viability envelope, it can develop freely. When it comes near the edge, however, and its viability begins to be impaired, then the growth of the subsystems driving it to non-viability is slowed or stopped temporarily. Other subsystems, triggered to act by the increased stress, adjust to bring the system as a whole back within the viability envelope. The driving subsystems are then re-enabled, and the cycle of growth and correction begins again.

Critically, this is only possible if the system is able to determine the direction of stress, and if stress caused by one system can be relieved by adjustment of another. For example, if a beam has become the wrong length due to the change of structure around it, the beam will experience tensile stress if it is too short and compressive stress if it is too long. If only the magnitude of error is measured, then the beam cannot know whether it should grow or shrink to reduce stress, but if the direction of the stress is measured then the appropriate corrective measure becomes obvious.

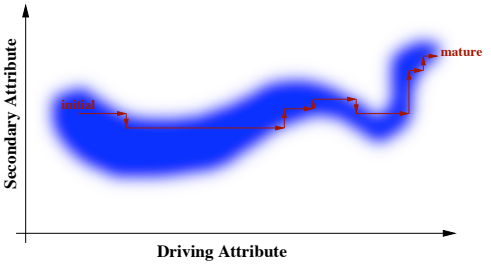


Fig. 1. In this abstract example, a growing system with two attributes uses stress tolerance to navigate through a complex viability envelope (blue). When unconstrained, the system grows its driving attribute (horizontal arrows). When the system's viability begins to be impaired (faint blue), it relieves that stress by adjusting its secondary attribute (vertical arrows). By repeatedly switching between driving growth and relieving stress, the system is able to navigate a complex viability envelope.

For example, consider an abstract system with two attributes, whose combination is viable only in the complex envelope shown in Figure 1. The horizontal attribute drives system growth, increasing whenever the system is clearly viable (horizontal arrows). When the system's viability begins to be impaired (faint blue), the secondary attribute adjusts to correct (vertical arrows). Given some hysteresis in the switch between driving and correction, the switch in modes need only occur a finite number of times. By repeatedly switching between driving growth and relieving stress, a system may navigate a complex viability envelope.

Thus we see that it is possible to use systemic stress as a signal to coordinate the growth of independently developing subsystems. This will not, of course, work for all possible such viability spaces: if the viability space includes a "dead end" that the driving attribute can push into, then it cannot be successfully navigated without additional guidance. For many systems, however, such as those where the coordination problem is rooted in the difference of scaling laws, (e.g. bone length (linear) vs. muscle cross-section (square) vs. lung capacity (cubic)), the viability space is guaranteed to be navigable. Note also that when stress is localized, the process of correction can be localized as well, allowing navigation to be parallelized when systems are not directly affecting one another. For example, different sets of developing muscles can be sore at the same time.

3 Functional Blueprints

Having made the observation that stress tolerance can allow a system to dynamically discover trajectories through its viability space, we can now take the next step and propose an engineering framework for predictably constructing such systems. Let us thus define a *functional blueprint* for some system X to consist of four elements:

1. A *system behavior* that degrades gracefully across some range of viability. Formally, if C_X is a manifold of possible configurations of system X, then it must be possible to establish a concave viability function v_X mapping $C_X \to [0, \infty)$ such that for any configuration c_X, only viable configurations have $v_X(c_X) > 0$,[1] and for any such configuration there exists a ball $B \in C_X$ centered on c_X such that $v_X(B) > 0$.
2. A *stress metric* quantifying the degree and direction of stress on the system. Formally, let the stress metric s_X be a vector field on C_X such that s_X is the gradient of some legal viability function for C_X.
3. An *incremental program* that relieves stress through growth (or possibly shrinking). Formally, let this be a parametrized map $i_{X,\epsilon,d} : C_X \to C_X$ that shifts a configuration by ϵ distance in the direction d.
4. A program to construct an initial *minimal system*. This initial minimal system, which we label X_0, must be viable ($v_X(X_0) > 0$).

Graceful degradation of system behavior asserts that the core functionality of the system must not have a sharp transition between viable and non-viable. The stress metric and incremental program combine to shift a degraded system's configuration back toward viability. Finally, the minimal system makes sure there is some viable place to start.

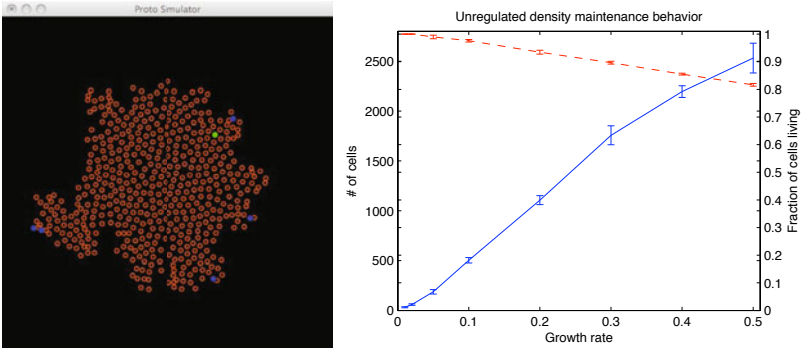
To transfer these properties to a composite system, it is necessary only to ensure that the subsystems are coupled such that the side effects of subsystems on one another are incremental. Formally, the action of each subsystem X's incremental program on each other subsystem Y forms a continuous map, $\pi_{X,Y}$. Given such a coupling of functional blueprints, it is always the case that it is possible to adjust any given subsystem by some small increment without knocking any other subsystem out of its range of viability. This can be proved by construction:

Theorem 1. *Consider a system S, for which every subsystem has a functional blueprint, and let X and Y be subsystems of S. For any given configuration c_S, if $v_X(c_X) > 0$, then there exists a $\delta > 0$ such that $c'_S = i_{Y,\epsilon,d}(c_S)$ has $v_X(c'_X) > 0$ for every d and $\epsilon \leq \delta$.*

Proof. By graceful degradation, we know that there exists a ball B centered on c_X such that every point $b \in B$ also has $v_X(b) > 0$. By the continuity of the coupling map $\pi_{X,Y}$, we know that the preimage of $\pi^{-1}_{X,Y}(B)$ is an open set. Being an open set, the preimage must contain some ball B' of radius δ around the configuration c_Y. By the definition of an incremental program, any configuration c'_S accessible via subsystem Y's incremental program $i_{Y,\epsilon,d}(c_S)$ is within the ball B' for $\epsilon \leq \delta$. Since B' is a subset of $\pi^{-1}_{X,Y}(B)$, c'_X must be within B and must therefore have $v_X(c'_X) > 0$. □

A simple growing composite system, such as the one illustrated in Figure 1, can thus be constructed simply by taking the composite stress to be the maximum

[1] Note that not all viable configurations need have $v_X(c_X) > 0$: the point is for the viability function to serve as a conservative guide for system growth, not to capture the precise boundary at which the system fails.



(a) (b)

Fig. 2. Density maintenance and growth: (a) shows an expanding sheet of cells, where blue cells are reproducing and green are dying. (b) shows system behavior with respect to the growth rate parameter p_d. At low p_d, the number of cells after 200s of growth (blue) is small, but at high p_d the fraction of cells surviving (red) begins to drop.

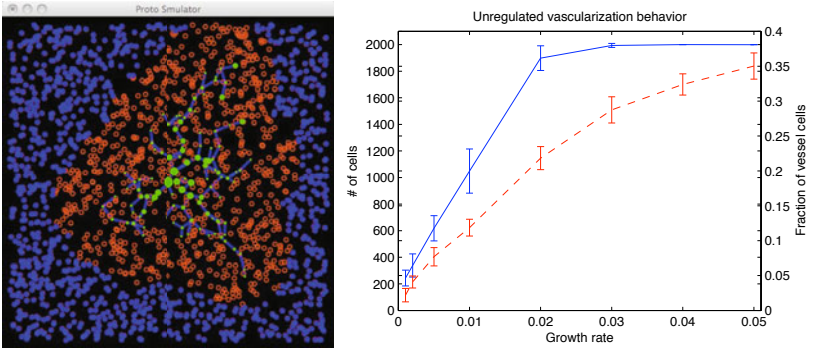
stress of any subsystem and executing the incremental program of the maximally stressed subsystem. The system can then be navigated toward a desired mature form by driving any subsystem or collection of subsystems whenever the composite stress is low enough.

4 Example Application: Tissue Growth

Having proposed a framework for the design of grown systems, let us now demonstrate its feasibility by developing a simplified model of tissue growth, in which the growth of a sheet of cells is synchronized with the growth of the blood vessels that supply them with oxygen. This example system should not be regarded as a serious model of tissue growth, but as a cartoon to demonstrate the feasibility of the engineering approach under discussion.

This simplified model consists of two subsystems, each specified with a functional blueprint. The cell density subsystem attempts to keep cells packed at a moderate density via motion, reproduction, and apoptosis. A consequence of this density maintenance is tissue growth at an approximately constant rate of expansion: cells at the surface of the tissue generally have a low average density, since there are no cells to one side of them, so unless they are regulated otherwise, they will tend to reproduce. The vascularization subsystem, on the other hand, attempts to ensure that no cell is too distant from a network conveying oxygenated blood outward from a source.[2] These two systems are linked together by adding a regulatory input to the cell density subsystem, such that cells will not attempt to reproduce if they are oxygen-starved. The resulting composite

[2] In this simplified system, venous return is not modeled, but could be implemented using a complementary mechanism.



(a) (b)

Fig. 3. Vascularization: (a) shows an expanding network (green dots, blue lines) expanding the area of oxygenation (red). The area of a dot is proportionally to the size of the network descending from it. (b) shows system behavior with respect to the growth rate parameter p_v. At low p_v, the oxygenated area (blue solid line) expands slowly, but at high p_v a large fraction of cells (red dashed line) are incorporated into vessels.

system produces smoothly synchronized tissue growth, and can be modulated to produce shaped tissue by external regulation of either subsystem.

These models are developed and simulated using the Proto spatial computing language [3], a functional language that allows the programmer to specify aggregate behaviors using scalable geometric descriptions. Aggregate behavior descriptions are then compiled into a program to execute on each cell (every cell is given the same program) and an interaction protocol by which cells cooperate to approximate the desired aggregate behavior. Details of Proto can be found in [3] in the MIT Proto distribution [10].

4.1 Cell Density

In this simplified model, the base structure and expansion of a sheet of cells is produced by a system that attempts to keep cells packed at a moderate density. We can implement such a system as follows:

```
(def cell-density (grow shrink p_d)
  (let ((packing (num-nbrs)))
    (clone (and grow (and (< packing 8) (< (rnd 0 1) p_d))))
    (die (or (and (> packing 15) (< (rnd 0 1) p_d))
             (and shrink (< (rnd 0 1) p_d)))))
  (disperse 0.6))
```

Here the desired system behavior is to maintain a moderate spacing between cells, which exhibits graceful degradation if the cells have some tolerance for overcrowding or underpopulation. The system is thus stressed when there are too many neighbors (here defined as more than 15), too few neighbors (here

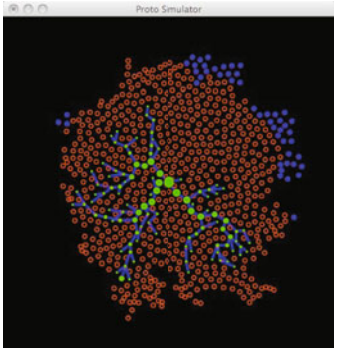


Fig. 4. Synchronized growth of a tissue: growth from cell density maintenance is enabled only for cells served by vascularization

defined as less than 8), or if the neighbors aren't at a desired separation (here defined as 0.6 communication radii).

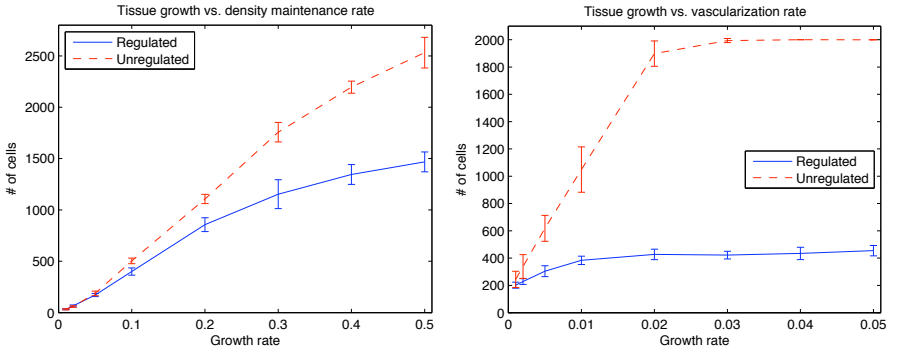
The incremental program relieves stress in a straightforward manner: when there are too many neighbors, the cell apoptoses (dies) with probability p_d, and when there are too few neighbors, the cell reproduces with probability p_d (the *grow* enabling input and *shrink* forcing input allows these actions to be modulated by an enclosing system). When the neighbors are not at a desired separation, they move towards it using spring forces:

```
(def disperse (packing)
  (* (/ 1 (int-hood 1))
     (int-hood (* (let ((dr (- (nbr-range) packing)))
                    (mux (< dr 0) dr (* 0.1 dr)))
                  (normalize (nbr-vec))))))
```

in which attractive forces are weaker than repulsive forces such that the farther neighbors do not exert too much influence and collapse the diameter of communicating clusters.

Note that since cells at the surface of the sheet have an expected density half that of cells in the interior of the sheet, their density will be considered too low (except in temporary high-density pockets) and they will reproduce. This has the desirable consequence of continually expanding the sheet of cells such that the edge moves outward at an expected constant rate.

Figure 2 shows this cell density system in experiments where the growth parameter p_d ranges from 0.01 to 0.5. For each parameter value, 10 trials were run, beginning with 10 cells distributed in a volume 10 units square and continuing for 200 simulated seconds. As the growth rate p_d rises, the final number of cells in the system (blue solid line) rises sharply, but the fraction of cells that die rises as well (red dashed line shows surviving cells). An intermediate level in the range 0.1 to 0.3 appears to offer the best trade-off, with graceful degradation as the parameter moves away from that level.



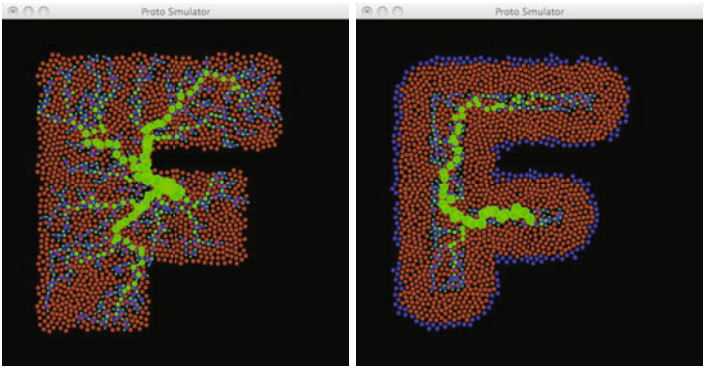
(a) (b)

Fig. 5. Linking density maintenance and vascularization results in synchronized tissue growth, with either subsystem able the regulate the behavior of the composite system

4.2 Vascularization

Oxygen delivery by a vascular system requires that there be a capillary vessel relatively close to every cell. When this is not the case, the cell becomes stressed by lack of sufficient oxygen—a graceful degradation situation since the cell does not die. The simple vascularization system here measures stress by distance to the nearest vessel:

```
(def vascularize (source service-range p_v)
  (rep (tup vessel served parent)
       (tup source source (if source (mid) -1))
       (mux source
         (tup 1 1 -1)
         (let ((service (< (gradient vessel) service-range))
               (server (gradcast vessel (mid)))
               (children (sum-hood (= (mid) (nbr parent))))
               (total-children (tree-children parent)))
           ;; adjust radius, for visualization
           (radius-set (mux vessel (* 0.5 (sqrt (+ 1 total-children))) 2))
           ;; grow/shrink vessel network
           (mux vessel
             (mux (or (muxand
                        (any-hood (and (= (nbr (mid)) parent)
                                       (> (nbr children) (mux (nbr source) 6 3))))
                        (not (any-hood (< (nbr total-children) total-children))))
                      (not (any-hood (and (nbr vessel) (= (nbr (mid)) parent)))))
               (tup 0 1 -1)    ; vessel is discarded
               (tup 1 1 parent)) ; vessels stay fixed
             (mux (muxand (muxand (any-hood (nbr vessel))
                                  (dilate (not served) service-range))
                          (< (rnd 0 1) p_v)))
               (tup 1 1 server)
               (tup 0 service -1)))))))
```



(a) (b)

Fig. 6. The tissue growth system can be modulated to produce complex patterns, such as the letter "F", by modulating growth of either the cell density (a) or vascularization subsystems (b). In both cases shown, the letter grows from a seed near its center.

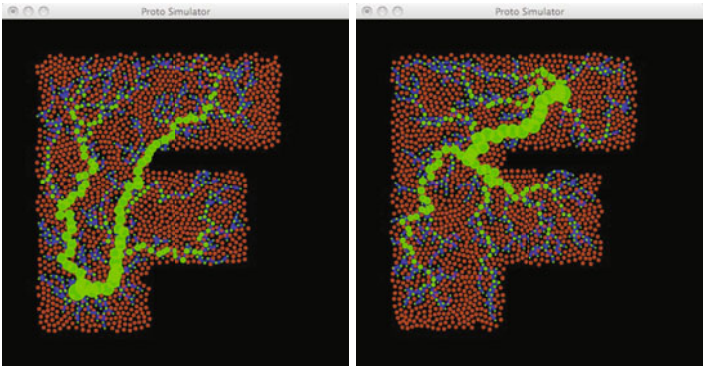
Every cell tracks whether it is part of a vessel and, if not, whether it has *service* from a vessel within the *service-range*, In the beginning, only the *source* cell(s) are part of a vessel. Later, the incremental program adds or removes cells from vessels to incrementally adjust the network. Cells join a vessel at a growth rate p_v when adjacent to a vessel and in range of an unserved cell. Vessel cells undifferentiate when they lose their connection to the source or when too many other vessel cells share the same junction.

Figure 3 shows the vascularization system in experiments where the growth parameter p_v ranges from 0.001 to 0.05. For each parameter value, 10 trials were run, where the network is grown for 200 simulated seconds from a seed point in the middle of a network of 2000 devices and devices are distributed on a square 300 by 300 units with a vascularization service range of 50 units. The higher the growth rate p_v, the faster that vascularization proceeds and therefore the larger an area that is served (blue solid line). The faster that vascularization proceeds, however, the more redundancy in the system, as reflected by the fraction of cells designated as vessels (red dashed line).

4.3 Composite Behavior

These two subsystems can be linked together into a simplified model of tissue growth by the simple expedient of enabling growth in the cell density system only for those cells served by vascularization:

```
(def tissue (src pd pv)
  (let ((v (vascularize src 50 pv)))
    (if (not src) (mov (cell-density (2nd v) 0 pd)) (tup 0 0 0))
    (drawvasc v)))
```



(a) (b)

Fig. 7. A functional blueprint separates the result of modulated tissue growth from the details of its execution, as shown by equivalent constructions of the letter "F" when grown from a seed in the lower right (a), upper left (b) or center (Figure 6(a))

Having implementing these two subsystems using functional blueprints, this simple coupling suffices for them to regulate one another into synchronized growth.

Figure 5 compares regulated behavior (blue line) with unregulated subsystems (red dashes), showing a smooth shift in regulatory dominance of the coupled system as p_d and p_v are varied. For each set of parameter values, 10 trials were run, beginning with 10 cells distributed in a volume 10 units square and continuing for 200 simulated seconds. In Figure 5(a), p_d is varied from 0.01 to 0.5 as above, while p_v is held constant at 0.02. At low values of p_d, growth from density maintenance dominates, but as p_d rises, cells spread outward faster and their growth begins to be checked by the rate of vascularization instead. In Figure 5(b), p_v is varied from 0.001 to 0.05 as above, while p_d is held constant at 0.1. At low values of p_v, vascularization is the limiting factor, but by $p_v = 0.02$ the limiting factory has shifted to the rate of growth from density maintenance.

This composite system may itself be viewed in terms of a functional blueprint, as these results illustrate, where both density and vascularization are being maintained in the face of stress, and the failure of either checks the other's progress. Moreover, just as the cell density subsystem was modulated to form a growing tissue, so may the tissue be modulated to grow complex shapes. This can be done by modulating either the cell density subsystem or the vascularization subsystem. For example, Figure 6 shows the result of constructing a letter "F" through regulating cell density (Figure 6(a)) and through regulating vascularization (Figure 6(b)).[3] Moreover, the functional blueprint separates the result of modulated tissue growth from the details of its execution, as illustrated by the equivalent constructions in Figure 6(a) and Figure 7.

[3] For simplicity in this demonstration, the "F" bounds are set by external localization, though it could be self-organized with variety methods (see [7], [9]).

5 Contributions

We have demonstrated that a *functional blueprint* approach can be used to create grown system that are dynamically integrated, smoothly transfer regulatory control across regimes, and can be interconnected to form composite systems with the same properties. While this is early work, the simplicity of creating and integrating the models discussed in this paper indicates good potential for further development. The decoupling of ultimate structure from developmental program might lead to more adaptivity in engineered systems as well as stronger biological models for evolvability and phenotypic adaptation.

References

1. Aubin, J.P.: Viability theory. Birkhäuser, Basel (1991)
2. Basu, S., Gerchman, Y., Collins, C.H., Arnold, F.H., Weiss, R.: A synthetic multicellular systems for programmed pattern formation. Nature 434, 1130–1134 (2005)
3. Beal, J., Bachrach, J.: Infrastructure for engineered emergence in sensor/actuator networks. IEEE Intelligent Systems 21(2), 10–19 (2006)
4. Carmeliet, P.: Angiogenesis in health and disease. Nat. Med. 9(6), 653–660 (2003)
5. Carroll, S.B.: Endless Forms Most Beautiful: The New Science of Evo Devo and the Making of the Animal Kingdom. W. W. Norton & Company (2005)
6. Coore, D.: Botanical Computing: A Developmental Approach to Generating Inter connect Topologies on an Amorphous Computer. Ph.D. thesis, MIT (1999)
7. Doursat, R.: The growing canvas of biological development: Multiscale pattern generation on an expanding lattice of gene regulatory networks. InterJournal: Complex Systems 1809 (2006)
8. Kirschner, M.W., Norton, J.C.: The Plausibility of Life: Resolving Darwin's Dilemma. Yale University Press, New Haven and London (2005)
9. Kondacs, A.: Biologically-inspired self-assembly of 2d shapes, using global-to-local compilation. In: 18th Int. Joint Conf. on Artificial Intelligence, pp. 633–638 (2003)
10. MIT Proto. Software available at `http://stpg.csail.mit.edu/proto.html` (Retrieved March 14, 2010)
11. Nagpal, R.: Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics. Ph.D. thesis, MIT (2001)
12. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. Springer, New York (1990)
13. Shetty, R.P., Endy, D., Thomas, F., Knight, J.: Engineering biobrick vectors from biobrick parts. Journal of Biological Engineering 2(5) (2008)
14. Spicher, A., Michel, O.: Declarative modeling of a neurulation-like process. BioSystems 87(2-3), 281–288 (2006)
15. Werfel, J., Ingber, D.E., Nagpal, R.: Collective construction of environmentally-adaptive structures. In: 2007 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 2345–2352 (2007)

Heterogeneous Particle Swarm Optimization

Andries P. Engelbrecht

Department of Computer Science, University of Pretoria, South Africa
engel@cs.up.ac.za

Abstract. Particles in the standard particle swarm optimization (PSO) algorithms, and most of its modifications, follow the same behaviours. That is, particles implement the same velocity and position update rules. This means that particles exhibit the same search characteristics. A heterogeneous PSO (HPSO) is proposed in this paper, where particles are allowed to follow different search behaviours selected from a behaviour pool, thereby efficiently addressing the exploration–exploitation trade-off problem. A preliminary empirical analysis is provided to show that much can be gained by using heterogeneous swarms.

1 Introduction

Particle swarm optimization (PSO) [3,8] is a stochastic, population-based optimization method. PSO algorithms maintain a *swarm* of candidate solutions, called *particles*. Each particle adjusts its position in search space by adding to its current position a step size, called the *velocity*. Step sizes are computed based on how far a particle is from the best position that the particle found during the search process, and how far the particle is from the best solution found by its neighborhood. The standard PSO and most of its modifications [4] make use of homogeneous swarms where all of the particles follow exactly the same behaviour. That is, particles implement the same velocity and position update rules. The effect is that particles have the same exploration and/or exploitation characteristics.

A very important aspect of optimization is the ability of an optimization algorithm to balance exploration and exploitation. Initially, the algorithm should focus on exploration, while preferring exploitation as the search process converges on an optimum. It is however difficult to determine at which point should the algorithm switch from an explorative behaviour to an exploitative behaviour. It may therefor be of an advantage to rather use heterogeneous swarms, where particles are allowed to implement different velocity and position update rules. By allowing particles to implement different update rules, a swarm may consist of explorative particles as well as exploitative particles. The optimization algorithm therefor has the ability to explore and exploit throughout the search process.

This paper proposes a heterogeneous PSO (HPSO), where particles in a swarm will be allocated different search behaviours by randomly selecting velocity and position update rules from a behaviour pool. The formal concept of heterogeneous swarms was introduced by Engelbrecht in [5], where the model investigated

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 191–202, 2010.

in this paper has been proposed. However, the idea of heterogenous swarms is not new. Examples of existing approaches where particles are allowed to implement different behaviours include, amongst others,

- The division of labor PSO [16], where particles are allowed to switch to a local search near the end of the search process.
- The life-cycle PSO [9], where particles follow a life-cycle, changing from a PSO particle, to a genetic algorithm individual, to a stochastic hill-climber. At any time, individuals may follow different behaviours.
- The predator-prey PSO [13], where the swarm contains predator and prey particles. Predator particles are attracted only to the global best position, thereby exploiting. Prey particles implement the standard PSO velocity update rule, but with an additional term added to repel prey particles from the position of predator particles.
- The guaranteed convergence PSO [15], where the global best particle follows a different, exploitative search behaviour than all the other particles.
- The NichePSO [2], developed to locate multiple solutions. A main swarm of particles is used, where particles implement a cognitive-only velocity update. Sub-swarms are formed around optima, with particles following the guaranteed convergence PSO.
- The charged PSO [1], where some particles have a charge and others not. Non-charged particles implement the standard velocity update rule, while charged particles add an additional repelling force to the velocity update rule.

Recent models that nake use of a more generic concept of heterogeneous behaviours include the heterogeneous cooperative algorithms developed by Olorunda and Engelbrecht [11], the heterogeneous PSO algorithms of Montes de Oca *et al* [10], and the adaptive heterogeneous PSO proposed by Spanevello and Montes de Oca [14]. The heterogeneous cooperative algorithm allows sub-swarms in a cooperative coevolutionary model to implement different meta-heuristic algorithms exhibiting different search behaviours. Montes de Oca *et al* considered different levels of heterogeneity, using the term update-rule heterogeneity to mean PSO algorithms where particles use different position and velocity update rules. In their work, ony two different update rules were used. Spanevello and Montes de Oca proposed that behaviours change during the optimization process.

The HPSO proposed in this paper differ from the above PSO algorithms, in that behaviours are randomly assigned from a pool of behaviours. Two strategies are proposed, one where the randomly selected behaviours remain static, and the other where behaviours change at each iteration by randomly selecting new behaviours from the behaviour pool. A preliminary empirical analysis is provided to show that the proposed HPSO has potential.

The rest of the paper is organized as follows: Section 2 discusses a few homogeneous PSO algorithms which differ only in the implemented position and velocity update rules. The HPSO is presented in Section 3, while some empirical results are provided in Section 4.

2 Homogeneous Particle Swarm Optimizers

This section provides a very compact overview of homogeneous PSO algorithms which will be used by the HPSO proposed in Section 3. PSO models are included that differ in their exploration and exploitation behaviours. Please note that this is not an extensive review of homogeneous PSO algorithms.

2.1 Traditional Position and Veclosity Updates

The traditional PSO particle velocity and position updates, assuming a star neighborhood topology, are given as follows [3,8]:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t)) + c_2 r_{2j}(t)(\hat{y}_j(t) - x_{ij}(t)) \quad (1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (2)$$

where $x_{ij}(t), y_{ij}(t)$ and $\hat{y}_j(t)$ refer respectively to particle i's position, personal best position, and global best position in dimension j at time step t. The constants c_1 and c_2 are the acceleration coefficients, and $r_{1j}(t), r_{2j}(t) \sim U(0,1)$. In the above, w is the inertia weight.

Equation (1) result in particles with a balance in exploration and exploitation depending on the values of the parameters, w, c_1 and c_2. For the purposes of this paper, c_1 is initially set to a value larger than c_2. Over time, c_1 is linearly decreased, while c_2 is linearly increased [12]. This will focus on exploration during the initial search steps, moving towards more exploitation as the number of iterations increases.

2.2 Cognitive-Only Model

The cognitive-only velocity update [6] removes the social component from equation (1), to result in

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t)) \quad (3)$$

The same position update as in equation (2) is used. The cognitive-only model results in more exploration, due to the fact that each particle becomes a hill-climber.

2.3 Social-Only Model

The social-only velocity update [6] removes the cognitive component from equation (1), to result in

$$v_{ij}(t+1) = wv_{ij}(t) + c_2 r_{2j}(t)(\hat{y}_j(t) - x_{ij}(t)) \quad (4)$$

The same position update as in equation (2) is used. The cognitive-only model results in faster exploitation, as the entire swarm is one stochastic hill-climber.

2.4 Barebones PSO

Kennedy [7] developed the barebones PSO, where the velocity update is replaced with

$$v_{ij}(t+1) \sim N\left(\frac{y_{ij}(t)+\hat{y}_j(t)}{2}, \sigma\right) \tag{5}$$

where $\sigma = |y_{ij}(t) - \hat{y}_j(t)|$. The position update changes to

$$x_{ij}(t+1) = v_{ij}(t+1) \tag{6}$$

Note that the velocity no longer serves as a step size, but is the actual new position of the particle, sampled from the above Gaussian distribution. The barebones PSO facilitates initial exploration, due to large deviations (initially, personal best positions will be far from the global best position). As the number of iterations increases, the deviation approaches zero, focussing on exploitation of the average of the personal best and global best positions.

2.5 Modified Barebones PSO

Kennedy [7] modified the barebones velocity equation to improve its exploration abilities. The new velocity update is

$$v_{ij}(t+1) = \begin{cases} y_{ij}(t) & \text{if } U(0,1) < 0.5 \\ N\left(\frac{y_{ij}(t)+\hat{y}_j(t)}{2}, \sigma\right) & \text{otherwise} \end{cases} \tag{7}$$

Exploration is increased during the initial stages of the search process by focusing 50% of the time on personal best positions. Recall that personal best positions will initially differ significantly due to uniform random initialization of particles. As the process converges, the focus will move to exploitation, as all personal best positions will converge towards the global best position.

3 Heterogeneous Particle Swarm Optimization

The HPSO proposed in this paper selects random behaviours for particles from a pool of behaviours. Each behaviour consists of a pair containing a position update and a velocity update. For the purposes of this paper, the pool of behaviours include the five models summarized above in Section 2. Two different HPSO models are proposed, namely

- The **static** HPSO (sHPSO), where behaviours are randomly assigned to particles during initialization. The assigned behaviours do not change during the search process.
- The **dynamic** HPSO (dHPSO), where paricle behaviours can randomly change during the search process. For the purposes of this paper, a particle randomly selects new behaviours from the behaviour pool when the particle fails to improve its personal best position over a window of recent

iterations. If the personal best position does not change, it may indicate early stagnation, which can be addressed by assigning a new search behaviour to the particle.

The only changes to the algorithm flow of the standard PSO are therefor a step to initialize the behaviours of particles, and for the dynamic HPSO, to assign new behaviours for particles that stagnate. More elaborate schemes to trigger reinitialization of behaviours can be implemented. However, as this is a preliminary study, additional mechanisms will be explored in future research.

4 Empirical Results

This section provides preliminary results to indicate that the HPSO model shows potential to be further explored. For the purposes of this paper, the two heterogeneous models were compared with the homogeneous models summarized in Section 2, in order to determine if any gain can be achieved with a heterogeneous model of these behaviours. Each algorithm is tested on the following functions:

- **Ackley**: $f(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2}} - e^{\frac{1}{n}\sum_{j=1}^{n}\cos(2\pi x_j)} + 20 + e$, where $x_j \in [-30, 30]$.
- **Quadric**: $f(\mathbf{x}) = \sum_{l=1}^{n}\left(\sum_{j=1}^{l} x_j\right)^2$, where $x_j \in [-100, 100]$.
- **Rastrigin**: $f(\mathbf{x}) = 10n + v\sum_{j=1}^{n}\left(x_j^2 - 10\cos(2\pi x_j)\right)$, where $x_j \in [-5.12, 5.12]$.
- **Rosenbrock**: $f(\mathbf{x}) = \sum_{j=1}^{n-1}\left(100(x_{j+1} - x_j^2)^2 + (x_j - 1)^2\right)$, where $x_j \in [-30, 30]$.
- **Salomon**: $f(\mathbf{x}) = v - \cos(2\pi\sum_{j=1}^{n} x_j^2) + 0.1\sqrt{\sum_{j=1}^{n} x_j^2} + 1$, where $x_j \in [-600, 600]$.
- **Griewank**: $f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{j=1}^{n} x_j^2 - \prod_{j=1}^{n}\cos\left(\frac{x_j}{\sqrt{j}}\right)$, where $x_j \in [-600, 600]$.

This results in two unimodal (Quadric and Rosenbrock) and four multimodal (Ackley, Rastrigin, Salomon, Griewank) functions. Note that Rosenbrock, Griewank, and Salomon are not separable.

The functions were used in 10, 30, 50, and 100 dimensions. Each algorithm was executed on each function for 30 independent runs of 1000 iterations. Swarm sizes of 50 particles were used. The inertia weight was set to $w = 0.72$ for all the velocity update rules. For the velocity update rule in equation (1), c_1 started at 2.5, linearly reduced to 0.5; c_2 started at 0.5, linearly increased to 2.5. For all other velocity updates, $c_1 = c_2 = 2.5$.

All algorithms were implemented using CIlib (http://www.cilib.net).

Tables 1 and 2 present the fitness of the best solution found at the end of the 1000 iterations, averaged over the 30 simulations, together with standard deviations. Table 3 summarizes these results by ranking the algorithms based on average best fitness. Figure 1 illustrates the performance of the algorithms over time for all functions in 50 dimensions, and figure 2 illustrates the scalability of all algortihms.

Table 1. Comparative results, showing average best solutions found for Ackley, Quadric, Rastrigin

Algorithm	Ackley	Quadric	Rastrigin
	10 Dimensions		
Standard PSO	2.25E+00±9.09E-01	2.17E+01±3.61E+01	1.67E+01±7.46E+00
Social PSO	3.27E+00±1.37E+00	6.92E+01±9.11E+01	1.85E+01±8.32E+00
Cognitive PSO	1.95E+01±4.24E-01	1.48E+04±3.56E+03	1.19E+02±9.88E+00
Barebones PSO	4.23E-15±9.01E-16	6.33E-19±1.71E-18	5.34E+00±3.14E+00
Modified Barebones	3.40E-15±0.00E+00	1.01E-07±1.90E-07	6.63E-02±2.52E-01
Static HPSO	3.99E-15±0.00E+00	1.34E-11±3.44E-11	1.47E+00±2.59E+00
Dynamic HPSO	4.44E-16±0.00E+00	2.07E-08±6.96E-08	2.02E+00±2.09E+00
	30 Dimensions		
Standard PSO	8.71E+00±9.52E-01	2.78E+03±1.20E+03	1.28E+02±2.81E+01
Social PSO	9.95E+00±1.49E+00	5.13E+03±2.32E+03	1.08E+02±2.08E+01
Cognitive PSO	2.04E+01±2.47E-01	1.28E+05±4.15E+04	4.32E+02±2.79E+01
Barebones PSO	2.05E-08±2.94E-08	8.55E+02±5.23E+02	6.02E+01±1.44E+01
Modified Barebones	4.10E-07±2.80E-07	7.10E+03±4.42E+03	1.51E+01±5.67E+00
Static HPSO	1.20E+00±7.79E-01	8.71E+00±1.66E+01	1.75E+01±2.85E+01
Dynamic HPSO	1.08E-10±1.64E-10	3.65E-01±1.03E+00	1.62E+00±6.35E+00
	50 Dimensions		
Standard PSO	1.01E+01±9.32E-01	9.89E+03±4.97E+03	2.81E+02±4.58E+01
Social PSO	1.17E+01±1.00E+00	1.34E+04±4.55E+03	2.82E+02±4.03E+01
Cognitive PSO	2.06E+01±1.12E-01	3.47E+05±1.03E+05	7.67E+02±4.12E+01
Barebones PSO	1.14E-01±3.57E-01	2.79E+04±1.05E+04	1.46E+02±2.43E+01
Modified Barebones	1.23E-02±5.18E-03	8.19E+04±3.63E+04	7.84E+01±3.77E+01
Static HPSO	2.87E+00±4.55E+00	1.29E+03±2.13E+03	4.47E+01±7.08E+01
Dynamic HPSO	4.65E-09±1.19E-08	2.47E+01±8.48E+01	6.64E-02±3.635E-01
	100 Dimensions		
Standard PSO	1.22E+01±8.25E-01	4.06E+04±1.45E+04	6.72E+02±5.14E+01
Social PSO	1.32E+01±7.83E-01	6.15E+04±2.53E+04	7.21E+02±6.61E+01
Cognitive PSO	2.08E+01±7.39E-02	1.42E+06±5.19E+05	1.62E+03±4.10E+01
Barebones PSO	1.20E+01±9.30E-01	2.70E+05±1.00E+05	1.30E+03±3.61E+02
Modified Barebones	9.38E+00±2.29E+00	5.77E+05±2.15E+05	5.81E+02±1.30E+02
Static HPSO	2.87E+00±4.55E+00	2.41E+04±3.87E+04	1.24E+02±1.93E+02
Dynamic HPSO	1.60E-07±7.13E-07	1.28E+01±2.28E+03	1.78E-12±5.69E-12

Figure 1 shows that dHPSO provided the best fitness for 50 dimensions over all of the functions. This is confirmed in Table 3, which shows an average rank of 1 for dHPSO over all functions. This observation of dHPSO's best performance is also for 100 dimensions. For 30 dimensions, dHPSO has the best rank of 1.5, due to 3 functions (Rastrigin, Rosenbrock, Griewank) where dHPSO was the second best performer (after sHPSO and the modified barebones PSO). For 10 dimensions dHPSO has the second best rank of 2.17 after the modified barebones PSO which obtained a rank of 2. The average rank over all dimensions for each function show that dHPSO has the best rank for all but one function (Rosenbrock), where sHPSO has the best rank. When considering the overall rank, as the average rank over all functions and all dimensions, the HPSO models performed best, with dHPSO having a rank of 1.42 followed by sHPSO with a

Table 2. Comparative results, showing average best solutions found for Rosenbrock, Salomon, Griewank

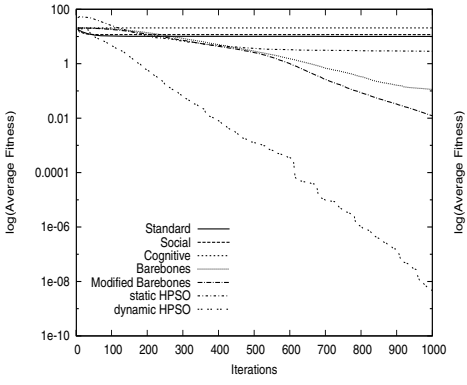
Algorithm	Rosenbrock	Salomon	Griewank
	10 Dimensions		
Standard PSO	7.39E+01±1.25E+02	2.01E+00±1.41E+00	3.33E-01±2.52E-01
Social PSO	2.16E+03±7.69E+03	3.09E+00±1.79E+00	4.40E-01±2.46E-01
Cognitive PSO	2.81E+07±1.58E+07	7.21E+01±8.22E+00	1.24E+02±3.28E+01
Barebones PSO	9.09E+00±1.72E+01	1.30E-01±4.66E-02	7.48E-02±4.25E-02
Modified Barebones	4.82E+00±6.19E+00	9.99E-02±1.12E-07	1.43E-02±1.55E-02
Static HPSO	1.89E+00±9.15E+00	1.47E-01±5.07E-02	7.82E-02±4.68E-02
Dynamic HPSO	4.91E+00±3.12E+00	4.99E-02±5.08E-02	5.59E-02±4.38E-02
	30 Dimensions		
Standard PSO	1.52E+05±9.99E+04	2.47E+01±4.09E+00	1.18E+01±3.73E+00
Social PSO	2.11E+05±1.43E+05	2.75E+01±4.40E+00	1.49E+01±4.21E+00
Cognitive PSO	2.28E+08±3.88E+07	1.55E+02±7.61E+00	5.79E+02±5.99E+01
Barebones PSO	7.19E+01±1.02E+02	4.13E-01±7.30E-02	8.12E-03±8.12E-03
Modified Barebones	8.83E+01±4.45E+01	4.02E-01±5.90E-02	7.52E-04±2.27E-03
Static HPSO	1.49E+01±3.89E+01	4.20E-01±8.86E-02	4.07E-02±1.29E-01
Dynamic HPSO	2.64E+01±4.15E-01	7.67E-02±4.28E-02	3.04E-03±7.17E-03
	50 Dimensions		
Standard PSO	6.52E+05±2.60E+05	4.08E+01±4.44E+00	3.67E+01±8.50E+00
Social PSO	1.21E+06±5.08E+05	4.63E+01±4.73E+00	4.50E+01±8.82E+00
Cognitive PSO	4.92E+08±5.55E+07	2.09E+02±7.78E+00	1.08E+03±8.55E+01
Barebones PSO	1.90E+06±1.04E+07	1.97E+00±4.60E-01	1.27E-02±1.93E-02
Modified Barebones	4.01E+02±3.03E+02	2.13E+00±3.66E-01	1.26E-02±1.39E-02
Static HPSO	5.25E+01±1.36E+02	1.31E+00±1.42E+00	1.54E-01±5.57E-01
Dynamic HPSO	4.67E+01±3.94E-01	7.33E-02±4.48E-02	6.65E-04±2.50E-03
	100 Dimensions		
Standard PSO	3.76E+06±2.21E+06	7.34E+01±5.53E+00	1.17E+02±2.85E+01
Social PSO	5.74E+06±1.69E+06	8.08E+01±6.65E+00	1.48E+02±1.68E+01
Cognitive PSO	1.13E+09±8.88E+07	3.11E+02±7.79E+00	2.39E+03±1.27E+02
Barebones PSO	1.15E+09±7.57E+07	2.97E+02±5.43E+01	2.18E+03±5.90E+02
Modified Barebones	4.71E+08±3.56E+08	4.38E+02±9.12E+00	4.39E+01±3.43E+01
Static HPSO	3.38E+03±7.70E+03	1.86E+01±7.94E+00	3.61E+00±4.08E+00
Dynamic HPSO	9.87E+01±5.92E+00	7.99E-02±4.04E-02	1.43E-07±7.84E-07

rank of 2.92. This is a clear illustration that, on average, the HPSO models, specifically the dHPSO, performed better than the homogenous models used within the HPSO models.

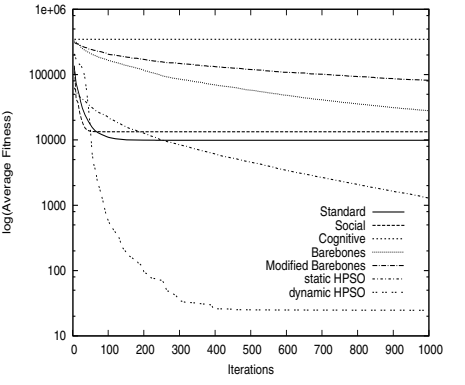
Figure 2 shows that the HPSO algorithms were by far the most scalable. In rank order from best scalable to worst scalable (ranks are given in parentheses): dHPSO (1), sHPSO (1.8), modified barebones PSO (3.5), standard gbest PSO (4.2), barebones PSO (5), social-only PSO (5.3), and cognitive-only (6.8). Important to note is that dHPSO was not significantly affected by an increase in dimensions, whereas sHPSO showed a small deterioration in performance compared to the other PSO algorithms. Note that, in general, the homogeneous PSO algorithms' performance deteriorate significantly as the number of dimensions increases.

Table 3. Performance Ranks for All Experiments; Ack (Ackley), Quad (Quadric), Ras (Rastrigin), Ros (Rosenbrock), Sal (Salomon), Grie (Griewank)

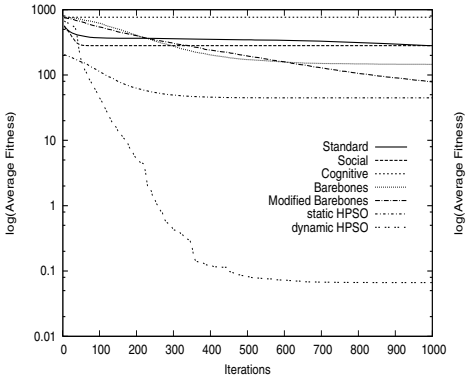
Algorithm	Ack	Quad	Ras	Ros	Sal	Grie	Average
	10 Dimensions						
Standard PSO	5	5	5	5	5	5	5
Social PSO	6	6	6	6	6	6	6
Cognitive PSO	7	7	7	7	7	7	7
Barebones PSO	4	1	4	4	3	3	3.17
Modified Barebones	2	4	1	2	2	1	2
Static HPSO	3	2	2	1	4	4	2.67
Dynamic HPSO	1	3	3	3	1	2	2.17
	30 Dimensions						
Standard PSO	5	4	4	5	5	5	4.67
Social PSO	6	5	3	6	6	6	5.33
Cognitive PSO	7	7	6	7	7	7	6.83
Barebones PSO	2	3	7	3	3	3	3.5
Modified Barebones	3	6	1	4	2	1	2.83
Static HPSO	4	2	5	1	4	4	3.33
Dynamic HPSO	1	1	2	2	1	2	1.5
	50 Dimensions						
Standard PSO	5	3	3	4	5	5	4.17
Social PSO	6	4	4	5	6	6	5.17
Cognitive PSO	7	7	6	7	7	7	6.83
Barebones PSO	3	5	2	6	3	3	3.67
Modified Barebones	2	6	7	3	4	2	4
Static HPSO	4	2	5	2	2	4	3.17
Dynamic HPSO	1	1	1	1	1	1	1
	100 Dimensions						
Standard PSO	5	3	3	3	3	4	3.5
Social PSO	6	4	4	4	4	5	4.5
Cognitive PSO	7	7	2	5	6	7	5.67
Barebones PSO	4	5	6	6	5	6	5.33
Modified Barebones	3	6	7	7	7	3	5.5
Static HPSO	2	2	5	2	2	2	2.5
Dynamic HPSO	1	1	1	1	1	1	1
	Average over all Dimensions						
Standard PSO	5	3.75	3.75	4.25	4.5	4.75	4.33
Social PSO	6	4.75	4.25	5.25	5.5	5.75	5.25
Cognitive PSO	7	7	5.25	6.5	6.75	7	6.58
Barebones PSO	3.25	3.5	4.75	4.75	3.5	3.75	3.92
Modified Barebones	2.5	5.5	4	4	3.75	1.75	3.58
Static HPSO	3.25	2	4.25	1.5	3	3.5	2.92
Dynamic HPSO	1	1.5	1.75	1.75	1	1.5	1.42



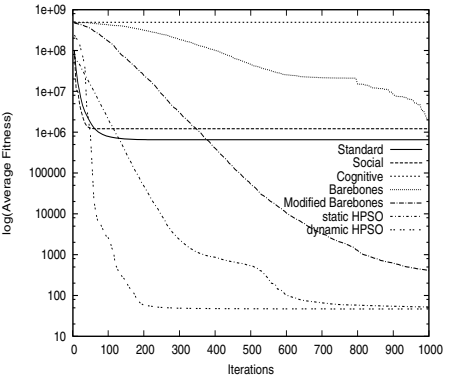
(a) Ackley



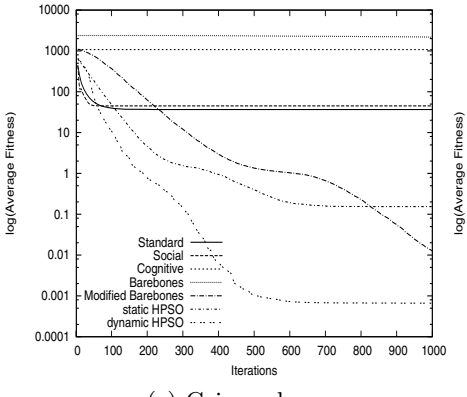
(b) Quadric



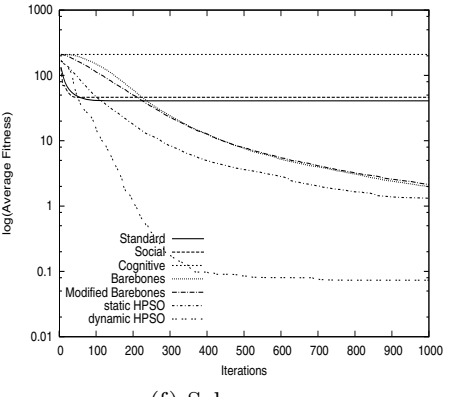
(c) Rastrigin



(d) Rosenbrock

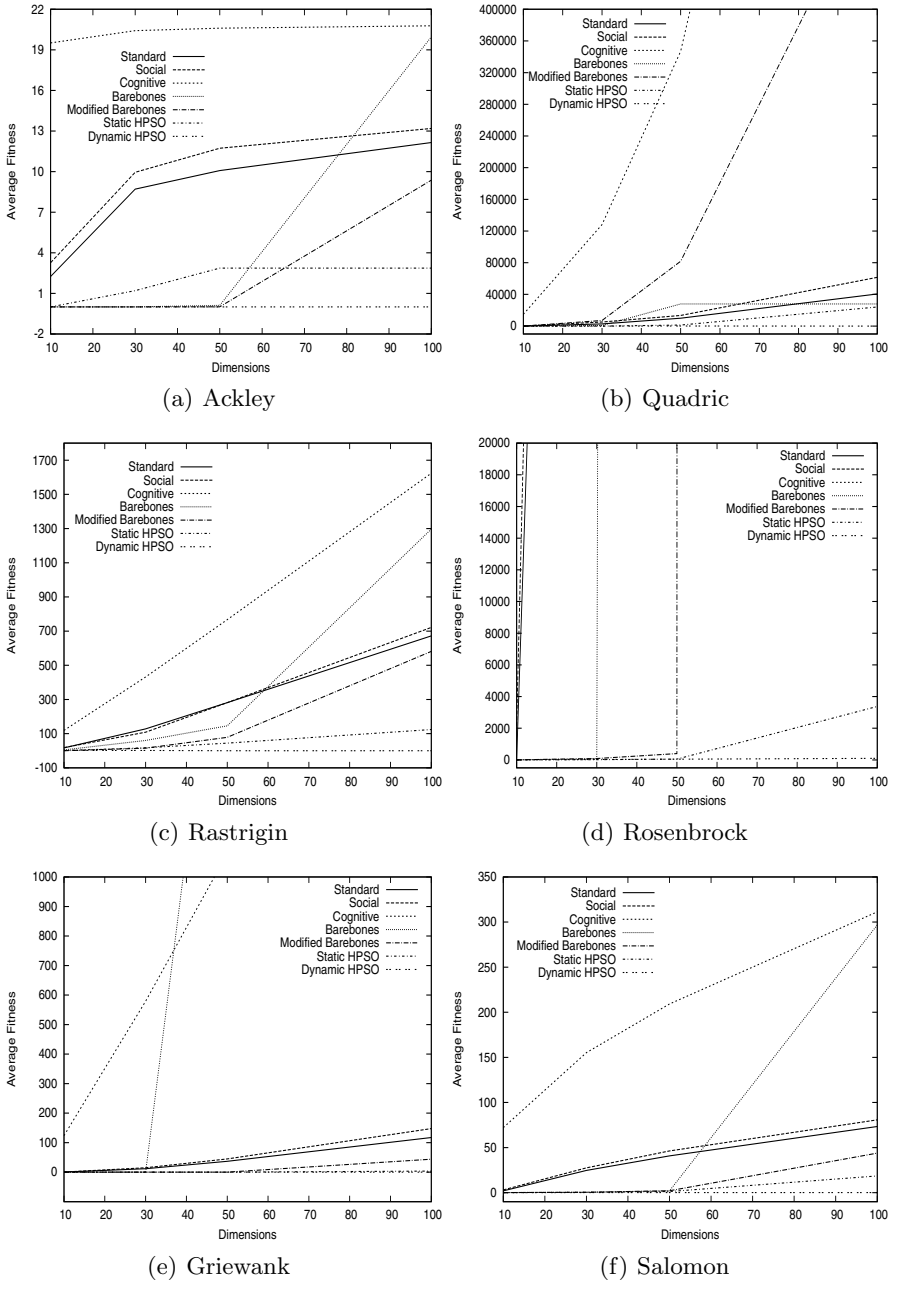


(e) Griewank



(f) Salomon

Fig. 1. Accuracy Profile for All Functions in 50 Dimensions



(a) Ackley (b) Quadric

(c) Rastrigin (d) Rosenbrock

(e) Griewank (f) Salomon

Fig. 2. Algorithm Scalability

5 Conclusions

This paper proposed a heterogeneous PSO (HPSO), where the particles of the swarm implements different behaviours in terms of the particle position and velocity updates. Two versions of the HPSO were proposed, the one static, where behaviours do not change, and a dynamic version where a particle may change its behaviour during the search process if it can not improve its personal best position. Both versions initialize particle behaviours by randomly selecting behaviours from a behaviour pool. When a particle in the dynamic HPSO has to change its behaviour, a new behaviour is randomly selected from the pool of behaviours.

For the purposes of this preliminary study, five different behaviours were included in the behaviour pool. These behaviours differ in the degree of exploration and exploitation, and how exploration is balanced with exploitation.

The empirical results have shown that the dynamic HPSO significantly outperformed the other two approaches in terms of the quality of the optima found and in terms of scalability. The static HPSO was the second best performer. This indicates that the HPSO has potential to be further explored. Future research will develop different models for behaviour change and will investigate other mechanisms to trigger a behaviour change.

References

1. Blackwell, T., Bentley, P.: Dynamic Search with Charged Swarms. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 19–26 (2002)
2. Brits, R., Engelbrecht, A., van den Bergh, F.: A Niching Particle Swarm Optimizer. In: Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning, pp. 692–696 (2002)
3. Eberhart, R., Kennedy, J.: A New Optimizer using Particle Swarm Theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science, pp. 39–43 (1995)
4. Engelbrecht, A.: Fundamentals of Computational Swarm Intelligence. Wiley & Sons, Chichester (2007)
5. Engelbrecht, A.: CIlib: A Component-based Framework for Plug-and-Simulate. In: International Conference on Hybrid Computational Intelligence Systems, Barcelona, Spain (2008) (Invites Talk)
6. Kennedy, J.: The Particle Swarm: Social Adaptation of Knowledge. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 303–308 (1997)
7. Kennedy, J.: Bare Bones Particle Swarms. In: Proceedings of the IEEE Swarm Intelligence Symposium, pp. 80–87 (2003)
8. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of the IEEE International Joint Conference on Neural Networks, pp. 1942–1948 (1995)
9. Krink, T., vberg, M.L.: The Life Cycle Model: Combining Particle Swarm Optimisation, Genetic Algorithms and Hill Climbers. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 621–630. Springer, Heidelberg (2002)

10. de Oca, M.M., Pena, J., Stuetzle, T., Pinciroli, C., Dorigo, M.: Heterogeneous Particle Swarm Optimizers. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 689–709 (2009)
11. Olorunda, O., Engelbrecht, A.: An Analysis of Heterogeneous Cooperative Algorithms. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1562–1569 (2009)
12. Ratnaweera, A., Halgamuge, S., Watson, H.: Particle Swarm Optimiser with Time Varying Acceleration Coefficients. In: Proceedings of the International Conference on Soft Computing and Intelligent Systems, pp. 240–255 (2002)
13. Silva, A., Neves, A., Costa, E.: An Empirical Comparison of Particle Swarm and Predator Prey Optimisation. In: O'Neill, M., Sutcliffe, R.F.E., Ryan, C., Eaton, M., Griffith, N.J.L. (eds.) AICS 2002. LNCS (LNAI), vol. 2464, pp. 103–110. Springer, Heidelberg (2002)
14. Spanevello, P., de Oca, M.M.: Experiments on Adaptive Heterogeneous PSO Algorithms. In: Proceedings of the Doctoral Symposium on Engineering Stochastic Local Search Algorithms, pp. 36–40 (2009)
15. van den Bergh, F., Engelbrecht, A.: A New Locally Convergent Particle Swarm Optimizer. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 96–101 (2002)
16. Vesterstrøm, J., Riget, J., Krink, T.: Division of Labor in Particle Swarm Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1570–1575. IEEE Press, Los Alamitos (2002)

Modern Continuous Optimization Algorithms for Tuning Real and Integer Algorithm Parameters

Zhi Yuan, Marco A. Montes de Oca, Mauro Birattari, and Thomas Stützle

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{zyuan,mmontes,mbiro,stuetzle}@ulb.ac.be

Abstract. To obtain peak performance from optimization algorithms, it is required to set appropriately their parameters. Frequently, algorithm parameters can take values from the set of real numbers, or from a large integer set. To tune this kind of parameters, it is interesting to apply state-of-the-art continuous optimization algorithms instead of using a tedious, and error-prone, hands-on approach. In this paper, we study the performance of several continuous optimization algorithms for the algorithm parameter tuning task. As case studies, we use a number of optimization algorithms from the swarm intelligence literature.

1 Introduction

Assigning appropriate values to the parameters of optimization algorithms is a task that frequently arises as part of the solution of application problems. This task has traditionally been tackled by software solution architects, who have knowledge about the application problem, but who do not necessarily have knowledge about the specific optimization algorithms employed. When optimization algorithm designers are involved, they have to learn many details about the application problem before suggesting a set of parameter values that they believe will provide good results. In any case, a significant amount of human effort is devoted to the solution of the parameter tuning problem [1].

Tackling algorithmically the parameter tuning problem [6] is of practical relevance because it offers the possibility of freeing software architects and designers from this time-consuming task. This can be done by casting the parameter tuning problem as an optimization problem, where the goal is to find parameter settings that optimize some performance statistics (e.g., the average performance) on typical instances of the application problem. Common performance measures are the solution quality reached after a specific computation time limit, or the computation time necessary for finding a solution of a specific quality level. A number of algorithms have been proposed for this task over the years. Several of these efforts focused on finding good values to numerical parameters, which resulted in methods such as CALIBRA [1], REVAC [16], SPO [5] and SPO$^+$ [12]. Methods for setting numerical as well as categorical parameters (e.g., the type of local search in a memetic algorithm) have also been proposed. Examples are F-Race and iterated F-Race [7,8], ParamILS [13], genetic programming [17,10], and gender-based genetic algorithms [2].

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 203–214, 2010.

While the above-mentioned methods have proved their potential, we believe that more effective configuration algorithms can be developed if the numerical part of the general algorithm configuration problem is treated as a stochastic continuous optimization problem. In this paper, we investigate the effectiveness of modern continuous optimization techniques for tackling the problem of setting numerical parameters of optimization algorithms. In particular, we tackle parameter tuning tasks that involve continuous parameters, such as the pheromone evaporation rate in an ant colony optimization (ACO) algorithm, and integer parameters, such as the population size in evolutionary algorithms. We treat integer parameters as "quasi-continuous", that is, we assume that the real-valued numbers given by continuous optimization techniques can be rounded to the nearest integer. This is a reasonable approach when the domain of the integer parameter is large.

The scale of the tuning tasks considered in this paper is small. We tackle problems with two to six numerical parameters. The reasons for limiting the number of parameters are the following. First, assuming that the parameter tuning problem is part of a larger algorithm configuration problem with the values of categorical parameters fixed, we would like to explore the effectiveness of modern continuous optimization techniques as sub-solvers for exploring the search space of the remaining numerical parameters. The categorical parameters, which are typical for many configuration tasks, would then be handled by another solver at a higher level. The goal would be to explore the domains of the continuous or quasi-continuous parameters using as few evaluations as possible. Second, there are several parameter tuning tasks that involve continuous or quasi-continuous parameters only. For example, if an algorithm is going to be used for tackling a new class of problem instances, or on a new application situation, it would be better to first tune the algorithm's continuous or quasi-continuous parameters before proceeding with the re-design of the algorithm.

This article is structured as follows. In the next section, we give an overview of the algorithms we chose as candidate tuning algorithms. In Section 3, we introduce the experimental setup and the benchmark domains on which we tested these algorithms. Results are described in Section 4 and we conclude in Section 5.

2 Tuning Algorithms

For the parameter tuning task, we selected state-of-the-art black-box continuous optimization algorithms developed in the mathematical programming and in the evolutionary computation communities. Since these algorithms were not explicitly designed for tackling stochastic problems, we enhanced them in order to let them handle noise. As a baseline for comparison, we included uniform random and iterated random sampling in our experiments.

2.1 Basic Algorithms

Uniform Random and Iterated Random Sampling (URS & IRS). The results obtained with URS and IRS are used as a baseline for evaluation. URS

explores the space of possible parameter settings uniformly at random. The obvious drawback of this approach is its inability to focus the search on promising regions of the search space. The method we refer to as IRS is the sampling part of Iterated F-Race as described in [8]. In IRS, the steps of solution generation, selection and refinement are executed iteratively. The solution generation step involves sampling from Gaussian distributions centered at promising solutions and with standard deviations that vary over time in order to search around the best-so-far solutions.

Bound Optimization by Quadratic Approximation (BOBYQA). BOBYQA [20] is a derivative-free optimization algorithm based on the trust region paradigm. It is an extension of the NEWUOA [19] algorithm that is able to deal with bound constraints. At each iteration, BOBYQA computes and minimizes a quadratic model that interpolates m automatically-generated points in the current trust region. Then, either the best-so-far solution, or the trust region radius is updated. The recommended number of points to compute the quadratic model is $m = 2d + 1$ [20], where d is the dimensionality of the search space. NEWUOA, and by extension BOBYQA, is considered to be a state-of-the-art continuous optimization technique [4]. The initial and final trust region radii, as well as a maximum number of function evaluations before termination are its parameters.

Mesh Adaptive Direct Search (MADS). In MADS [3], a number of trial points lying on a *mesh* are generated and evaluated around the best-so-far solution at every iteration. If a new better solution is found, the next iteration begins with a possibly coarser mesh. If a better solution is not found after this first step, trial points from a refined mesh are generated and evaluated. In this second step, MADS differs from the generalized pattern search class of algorithms [26] in that MADS allows a more flexible exploration of this refined mesh by allowing the sampling of points at different distances from the best-so-far solution. When a new best solution is found, the algorithm iterates.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES). In CMA-ES [11], candidate solutions are sampled at each iteration from a multivariate Gaussian distribution. The main characteristic of CMA-ES is that the parameters of this distribution are adapted as the optimization process progresses. The mean of the sampling distribution is centered at a linear combination of the current "parent" population. The covariance matrix is updated using information from the trajectory the best solutions have followed so far. The aim of this transformation is to increase the chances of sampling improving solutions. CMA-ES is considered to be a state-of-the-art evolutionary algorithm [4].

2.2 Enhancing Noise Tolerance

The problem of tuning the parameters of an optimization algorithm can be seen as a stochastic optimization problem. The sources of stochasticity are the randomized nature of the algorithm itself and the "sampling" of the problem instances. We enhanced the algorithms described above with mechanisms for

better estimating the real difference between two or more candidate solutions. These mechanisms are described below.

Repeated Evaluation. The simplest approach to deal with noise in the evaluation of an objective function is to evaluate it more than once and return the average evaluation as the closest estimate of the true value. We denote by nr the number of times the objective function evaluation is repeated. The advantages of this approach are its simplicity and the confidence that can be associated with the estimate as a function of nr. We tried this approach with all methods using different values for nr. The main disadvantage of this technique is that it is blind to the actual quality of the solutions being re-evaluated, and thus many function evaluations can be wasted.

F-Race. It is a technique aimed at making a more efficient use of computational power than repeated evaluation in the presence of noise. Given a set of candidate solutions and a noisy objective function, the goal of F-Race is to discard those solutions for which sufficient statistical evidence against them has been gathered. The elimination process continues until one solution remains, or the maximum number of evaluations is reached. The elimination mechanism of F-Race is independent of the composition of the initial set of candidate solutions. It is thus possible to integrate it with any method that needs to select the best solutions from a given set. For this reason, F-Race is used with URS, IRS, MADS, and CMA-ES. F-Race is not used with BOBYQA because this algorithm generates one single trial point per iteration and does not need to select the best out of a set. More information about F-Race can be found in [7,8].

3 Benchmark Tuning Problems

We compare the performance of the continuous optimization algorithms described in Section 2 on six benchmark (parameter) *tuning problems*. Each *tuning problem* consists of a parameterized algorithm to be tuned, and an optimization problem to which this algorithm is applied. The six tuning problems are originated from three classes of case studies with three underlying algorithms to be tuned. Two of them are swarm intelligence algorithms, ACO and particle swarm optimization (PSO), while the other differential evolution (DE) is an evolutionary algorithm. ACO is used to tackle a combinatorial optimization problem. PSO and DE are used to tackle a continuous optimization problem. The three case studies are the following.

***MAX–MIN*. Ant System - Traveling Salesman Problem (MMASTSP).** $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [24] is one of the most successful ACO [9] algorithms and we consider here the following parameters. The weight of the pheromone information α and the heuristic information β, the pheromone evaporation rate ρ, and the number of ants m. Moreover, in $\mathcal{MM}$AS, the ratio γ between the maximum and minimum pheromone trail values is also of importance. Here, we tackle the well-known traveling salesman problem (TSP). For MMASTSP, an extra parameter is nn, which gives the

Table 1. Range and default value of each parameter considered for tuning MMASTSP with 2, 4, and 6 parameters

MMASTSP-2			MMASTSP-4			MMASTSP-6		
param.	range	def.	param.	range	def.	param.	range	def.
α	$[0.0, 5.0]$	1.0	ρ	$[0.0, 1.00]$	0.5	γ	$[0.01, 5.00]$	2.0
β	$[0.0, 10.0]$	2.0	m	$[1, 1200]$	25	nn	$[5, 100]$	25

Table 2. Range and default value of each parameter considered for tuning DE with 3 parameters (left) and PSO with 2 and 5 parameters (right)

DE-3		
param.	range	def.
N	$[4, 1000]$	1000
xo	$[0.0, 1.0]$	0.9
s	$[0.0, 1.0]$	0.8

PSO-2			PSO-5		
param.	range	def.	param.	range	def.
χ	$[0.0, 1.0]$	0.729	N	$[4, 1000]$	30
ϕ_1	$[0.0, 4.0]$	2.05	p	$[0.0, 1.0]$	1
			ϕ_2	$[0.0, 4.0]$	2.05

number of nearest neighbors considered in the solution construction. No local search is applied. We extract three tuning problems from MMASTSP with 2, 4, and 6 parameters, namely MMASTSP-2 (with α and β), MMASTSP-4 (plus m and ρ), and MMASTSP-6 (plus γ and nn). This is done by fixing the unused parameters to their default values (see Table 1). For the default values we follow the ACOTSP software [23].

We used the DIMACS instance generator [14] to create Euclidean TSP instances with 750 nodes, where the nodes are uniformly distributed in a square of side length 10 000. 1000 such instances are generated for the tuning process, and 300 for the testing process.

Differential Evolution - Rastrigin Function. DE [22] is a population-based, stochastic continuous optimization method that generates new candidate solutions by exploiting the spatial distribution of existing ones. In the so-called *DE/rand/S/bin* variant (the most common version is the one with $S = 1$) [25], there are $S + 2$ parameters to set: the population size N, the crossover probability xo, and the value of the scaling factor s. In this variant, the population size must be at least equal to $2S + 2$. We refer the reader to the left columns of Table 2 for the parameter ranges and the default values used in our experiments, and to [22,25] for more information about DE and its parameters. The default parameter settings for DE were those suggested in [21].

The experiments are carried out on problems derived from the Rastrigin function, each of which has different fitness distance correlation (FDC) [15]. The Rastrigin function, whose n-dimensional formulation is $nA + \sum_{i=1}^{n} (x_i^2 - A\cos(\omega x_i))$, can be thought of as a parabola with a superimposed sinusoidal wave with an amplitude and frequency controlled by parameters A and ω respectively. By changing the values of A and ω, one can obtain a whole family of problems. In our experiments, we set $\omega = 2\pi$, and we vary the amplitude A to obtain functions with

different FDCs. This was done by sampling the values of A from a normal distribution with mean equal to 10.60171 and standard deviation equal to 2.75. These values approximately map to a normally distributed FDC with mean equal to 0.7 and standard deviation equal to 0.1. The FDC was estimated using 10^4 uniformly distributed random samples over the search range. Other settings are the search range and the dimensionality, n, of the problem, which we set to $[-5.12, 5.12]^n$ and $n = 100$, respectively.

Particle Swarm Optimization - Rastrigin Function. We use a PSO [18] algorithm with two and five parameters. In the first case, the two parameters are the so-called constriction factor χ and a value assigned to both acceleration coefficients ϕ_1. In the second case, the free parameters are the population size N, the constriction factor χ, the value of each acceleration coefficient, ϕ_1 and ϕ_2, and a probability p of connecting any two particles in the population topology. For more information about PSO and its parameters, we refer the reader to [18]. More details about the parameters used in our experiments are listed in the right two columns of Table 2. For the default value of the parameters in PSO we follow [18]. We tackle the same family of Rastrigin functions as in the DE experiments, and generate 1000 instances for the tuning process and 1000 for the testing process.

4 Experiments

4.1 Experimental Setup

For each of the six benchmark tuning problems, 10 `trials` were run. Each `trial` is the execution of the `tuning process` together with a subsequent `testing process`. In the testing process, the final parameter setting returned by the tuning process is evaluated on a set of test instances. For the purpose of reducing experimental variance, in each trial of the tuning process, we use a fixed random order of the tuning instances, and each instance is evaluated with a common random seed. The comparison of sampling algorithms is done across six benchmark tuning problems using the pairwise Wilcoxon signed rank test with blocking on each instance and Holm's adjustment for multiple test correction.

For each sampling algorithm, four levels of nr, that is, the number of times the optimization algorithm being tuned is executed with the same parameter settings, are considered: $5, 10, 20, 40$. We also consider for each tuning problem four different *tuning budgets*, that is, the maximum number of times that the target algorithm can be run during the tuning process. Let d be the dimensionality of the tuning problem, that is, the number of parameters to be tuned. The first and minimum level of the tuning budget is chosen to be $B_1 = 40 \cdot (2d + 2)$, e.g. $B_1 = 240$ when $d = 2$. The setting of B_1 is chosen in this way since BOBYQA needs at least $2n + 1$ points to make the first quadratic interpolation, and this setting guarantees that BOBYQA with $nr = 40$ can make at least one quadratic interpolation guess. The rest of the three levels of tuning budgets double the previous level respectively, that is, $B_i = 2^{i-1} \cdot B_1, i = 2, 3, 4$.

In our experiments, all parameters are tuned with a 2 significant digits precision. This was done by rounding each sampled value. This choice was made because we observed during experimentation that the lower the number of significant digits, the higher the performance of the tuned parameters. In each trial, the historical evaluation results are stored in an archive list, so that if the same algorithm configuration is sampled twice, the results on the evaluated instances will be read from the archive list without re-evaluating.

4.2 Settings of the Sampling Algorithms

In our experiments, the sampling algorithms, which generate the trial configurations, that is, algorithms such as CMA-ES, MADS and BOBYQA, are extended by a restart mechanism that is triggered whenever stagnation behavior is detected. Stagnation can be detected when the search coarseness of the mesh or the trust region radius drop to a very low level; for example, less than the degree of the significant digit. Each restart best solution is stored, and in the post-execution phase the best across all restart best solutions is selected by `F-Race`. The tuning budget reserved for the post-execution `F-Race` is determined by a factor μ_{post} times the number of restart best solutions. The factor μ_{post} in the repeated evaluation experiments is determined by $\mu_{post} = max\{5, (20-nr)\}$, where nr is the number of repeated evaluations. Also in the post-execution `F-Race`, we start the Friedman test for discarding candidates from the $max\{10, nr\}$-th instance, instead of five as in the normal `F-Race` setting to make the selection more conservative. Based on our experiments, the hybrid with `F-Race` is indeed better performing than its counterpart, and the advantage becomes statistically significant in the case of low nr value ($nr = 5$).

For the execution of each sampling algorithm, the default settings are adopted in our experiments, except CMA-ES. Due to the small number of sampled points, we modify CMA-ES by applying a uniform random sampling in the first iteration, where the best point will serve as a starting point for CMA-ES. This modification results in significant improvements for most of the case studies, especially when the number of sampled points is small.

We first study the setting of nr for all sampling algorithms. The statistical results unanimously show that for each sampling algorithm, the smaller the value of nr, the better performance is obtained, that is, $nr = 5$ is the best setting. Furthermore, all pairwise comparisons show statistically significant differences.

We consider also use of `F-Race` during the execution of the sampling algorithms instead of the fixed number of repeated evaluations. For IRS and URS, we adopted the iterated `F-Race` and random sampling `F-Race` from [8]. For `MADS/F-Race` we have adopted the algorithm of [27] by allowing the budget for each `F-Race` to be 10 times the number of candidate configurations. CMA-ES uses a (μ, λ)-ES, that is, μ elite configurations are used to generate λ new candidate configurations of the next iteration. Therefore, for `CMAES/F-Race` the iteration best configurations are stored, and the best configuration will be selected in a post-execution from the set of iteration best configurations by `F-Race`.

The post-execution `F-Race` is performed in the same way described above as for restart best configurations, except that we set $\mu_{post} = 10$.

Given that a setting of $nr = 5$ resulted in best performance, we directly compared the algorithms using this setting to the variants that were using F-Race for the selection of the best candidates. Pursuing the same analysis, we found that for two algorithms, IRS and URS, the versions with F-Race perform substantially better than the fixed sample size of $nr = 5$, while on CMA-ES and MADS the setting $nr = 5$ perform slightly, but statistically significantly better than the F-Race variants. Note that for MADS/F-Race the observation here contradicts the conclusions in [27]; a reason may be that here a restart version of MADS is used, while in [27] restarts are not considered.

Given this overall result for the comparison between the variants with F-Race and $nr = 5$ and the fact that BOBYQA is applied only with a fixed sample size, in the following analysis we focus on the case of $nr = 5$ only.

4.3 Comparisons of Continuous Optimization Algorithms

Here, we compare the performance of the five continuous optimization algorithms for the setting of $nr = 5$. Figure 1 shows the average costs of each of the five continuous optimization algorithms across four different tuning budgets. Each of the six plots shows the results for one tuning problem.

The main conclusions we obtain from this analysis is the following. For very small dimensional tuning problems with two and three parameters to be tuned, BOBYQA is the best performing algorithm. This is the case for the case studies MMASTSP2, PSO2, and DE3. However, BOBYQA's performance degrades very rapidly as the problem dimensionality increases. In PSO5, for example, it even performs worse than URS. The performance of CMA-ES is relatively robust across the various dimensionalities of the tuning problems and across the budget levels tested. The average ranking of CMA-ES among five algorithms is 2.29, substantially better than IRS and MADS (tied as 2.83). Finally, all the continuous optimization algorithms tested clearly outperformed URS in almost all tuning problems. (The same also holds if URS is combined with F-Race.)

Most of the differences that can be observed in Figure 1 are actually statistically significant. In Table 3 we indicate for each tuning problem and for each level of the tuning budget the ranking of the algorithms (from best to worst). All differences between consecutive pairs of algorithms are statistically significant except those where two algorithms are connected by a line (above or below the identifiers). Finally, Figure 2 shows the average ranking of the five algorithms grouped by the dimensionality of the tuning problem averaged across all budget levels. This figure confirms the observation that BOBYQA is the best for low dimensional tasks while CMA-ES shows rather robust performance.

4.4 Comparison between the Tuned and Default Configurations

We finally compared the results obtained with the tuned parameter configurations with those obtained with the default parameter configurations. Please refer

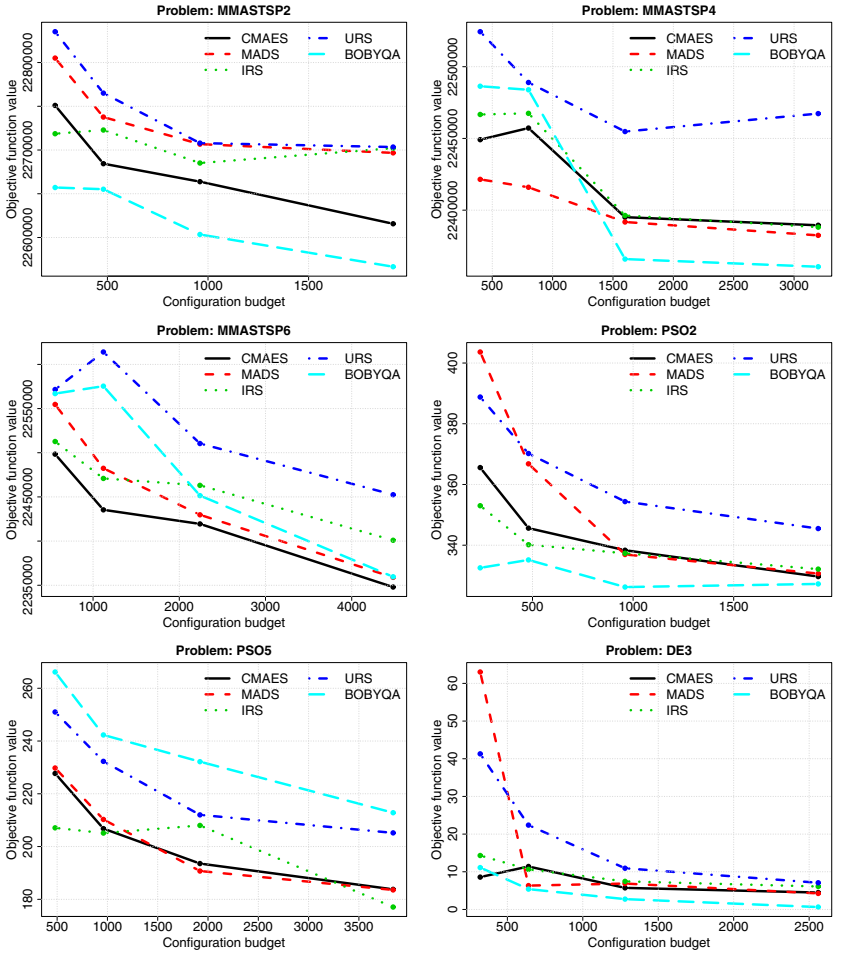


Fig. 1. The comparisons of different algorithms with $nr = 5$ on the six tuning problems

to Table 1 and 2 for the default parameter settings of the three case studies. The tuned parameter configurations strongly outperform the default configurations in all cases. Throughout the experiments, the default parameter configuration is only comparable with the tuned configuration in PSO2 with the lowest level budget (240). In the case study of MMASTSP, the tuned configurations improve, on average, over the default configuration by more than 10%. In the PSO case study, the tuned parameters improve, on average, over the default configuration by more than 30% in the PSO-2, and by more than 50% in PSO-5. The strongest improvement by tuning is observed in the case study of DE. The average cost obtained by the default configuration is 2168, which is one order of magnitude worse than the costs obtained by the worst tuned configurations using the smallest tuning budget. In fact, almost all tuning algorithms can find configurations

Table 3. Algorithms (using $nr = 5$) are ordered according to the ranking (form best to worst) for each tuning problem and for each budget level. The abbreviations used are B for BOBYQA, C for CMA-ES, I for IRS, M for MADS, and R for URS. The budget is from B_1 (low) to B_4 (high). In the ordering, an overline or an underline indicates that between these algorithms no statistically significant differences were observed.

budget	MMAS-2	MMAS-4	MMAS-6	DE-3	PSO-2	PSO-5
B_1	B I C M R	M C I B R	C I M B R	C B I R M	B I C R M	I C M R B
B_2	B C I M R	M C I B R	C I M B R	B M I C R	B I C M R	I C M R B
B_3	B C I M R	B M C I R	C M B I R	B C M I R	B M I C R	M C I R B
B_4	B C M I R	B M I C R	C M B I R	B M C I R	B C M I R	I M C R B

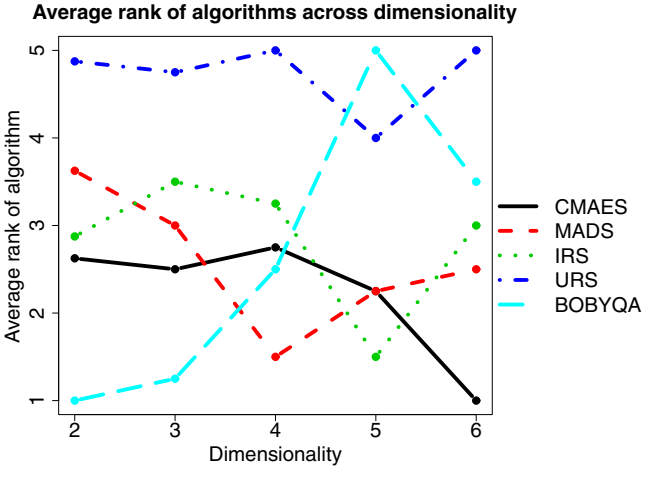


Fig. 2. The average rank of the sampling algorithms across dimensionalities of the tuning problems

that give results close to the optimal value 0. Furthermore, by the default parameter configuration, DE gives worse results than PSO (2168 vs. 589). However, the tuned DE performs much better than the tuned PSO (0.64 vs. 177). On the one side this shows that DE is very sensitive to its parameter settings. On the other side, this also proves that the algorithm tuning procedure can exploit the full potential of the algorithm, and should be applied before algorithm comparisons. These results confirm the practical importance of automated algorithm tuning in deriving high-performing algorithms.

5 Conclusions

In this paper, we compare the performance of three modern state-of-the-art continuous optimization algorithms, CMA-ES, BOBYQA and MADS, together

with the population-based URS and IRS, for the automatic tuning of numerical parameters. Two swarm intelligence algorithms, an ACO algorithm applied to the TSP and a PSO algorithm are considered as case studies, together with a DE algorithm. The sampling algorithms are improved by a restart mechanism, and CMA-ES is hybridized with a uniform random sampling in the first iteration.

The experiments show that, among the five continuous optimization algorithms, BOBYQA performs the best in low dimensional problems with two or three parameter to be set, but that it performs poorly on the case studies with more parameters to be set. CMA-ES appears to be a rather robust algorithm across all dimensionalities. In future work, we want to integrate continuous optimization algorithms with configuration methods that work on categorical parameters. In fact, considering a hybrid solver, where iteratively continuous tuning tasks arise that are then tackled by effective algorithms specialized to such problems, may be an interesting way to go to improve the performance of automated configuration algorithms also on more complex configuration tasks.

Acknowledgments. We thank Michael J. D. Powell for providing the Fortran code of BOBYQA. This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium. Zhi Yuan, Mauro Birattari, and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS.

References

1. Adenso-Díaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental designs and local search. Operations Research 54(1), 99–114 (2006)
2. Ansotegui Gil, C., Sellmann, M., Tierney, K.: A gender-based genetic algorithm for the automatic configuration of solvers. In: Gent, I.P. (ed.) CP 2009. LNCS, vol. 5732, pp. 142–157. Springer, Heidelberg (2009)
3. Audet, C., Dennis, J.E., Mesh, J.: adaptive direct search algorithms for constrained optimization. SIAM Journal on Optimization 17(1), 188–217 (2006)
4. Auger, A., Hansen, N., Zerpa, J.M.P., Ros, R., Schoenauer, M.: Experimental comparisons of derivative free optimization algorithms. In: SEA 2009. LNCS, vol. 5526, pp. 3–15. Springer, Heidelberg (2009)
5. Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation–The New Experimentalism. Springer, Berlin (2006)
6. Birattari, M.: The Problem of Tuning Metaheuristics as seen from a Machine Learning Perspective. Ph.D. thesis, Université Libre de Bruxelles (2004)
7. Birattari, M.: Tuning Metaheuristics: A machine learning perspective. Springer, Berlin (2009)
8. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-Race and iterated F-Race: An overview. In: Bartz-Beielstein, T., et al. (eds.) Experimental Methods for the Analysis of Optimization Algorithms, pp. 311–336. Springer, Berlin (2009)
9. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
10. Fukunaga, A.S.: Automated discovery of local search heuristics for satisfiability testing. Evolutionary Computation 16(1), 31–61 (2008)

11. Hansen, N.: The CMA evolution strategy: a comparing review. In: Lozano, J., et al. (eds.) Towards a new evolutionary computation, pp. 75–102. Springer, Berlin (2006)
12. Hutter, F., Hoos, H.H., Leyton-Brown, K., Murphy, K.P.: An experimental investigation of model-based parameter optimisation: SPO and beyond. In: Proc. of GECCO 2009, pp. 271–278. ACM press, New York (2009)
13. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research 36, 267–306 (2009)
14. Johnson, D.S., McGeoch, L.A., Rego, C., Glover, F.: 8th DIMACS implementation challenge, http://www.research.att.com/~dsj/chtsp/
15. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Proc. of 6th Int. Conf. on Genetic Algorithms, pp. 184–192. Morgan Kaufmann, San Francisco (1995)
16. Nannen, V., Eiben, A.E.: Relevance estimation and value calibration of evolutionary algorithm parameters. In: Proc. of IJCAI 2007, pp. 975–980 (2007)
17. Oltean, M.: Evolving evolutionary algorithms using linear genetic programming. Evolutionary Computation 13(3), 387–410 (2005)
18. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. An overview. Swarm Intelligence 1(1), 33–57 (2007)
19. Powell, M.J.D.: The NEWUOA software for unconstrained optimization. In: Large-Scale Nonlinear Optimization, Nonconvex Optimization and Its Applications, vol. 83, pp. 255–297. Springer, Berlin (2006)
20. Powell, M.J.D.: The BOBYQA algorithm for bound constrained optimization without derivatives. Tech. Rep. NA2009/06, Department of Applied Mathematics and Theoretical Physics, University of Cambridge (2009)
21. Storn, R.: Differential evolution homepage, http://www.icsi.berkeley.edu/~storn/code.html#prac
22. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4), 341–359 (1997)
23. Stützle, T.: Software ACOTSP, http://iridia.ulb.ac.be/~mdorigo/ACO/aco-code/public-software.html
24. Stützle, T., Hoos, H.: $\mathcal{MAX}$–$\mathcal{MIN}$. Ant System. Future Generation Computer Systems 16(8), 889–914 (2000)
25. Ting, C.K., Huang, C.H.: Varying number of difference vectors in differential evolution. In: Proc. of CEC 2009, pp. 1351–1358. IEEE Press, Piscataway (2009)
26. Torczon, V.: On the convergence of pattern search algorithms. SIAM Journal on Optimization 7(1), 1–25 (1997)
27. Yuan, Z., Stützle, T., Birattari, M.: MADS/F-race: mesh adaptive direct search meets F-race. In: Ali, M., et al. (eds.) Trends in Applied Intelligent Systems. LNCS, vol. 6096, pp. 41–50. Springer, Heidelberg (2010)

Multi-agent Deployment on a Ring Graph

Yotam Elor and Alfred M. Bruckstein

Faculty of Computer Science and the Goldstein UAV and Satellite Center, Israel
{yotame,freddy}@cs.technion.ac.il

Abstract. We consider two variants of the task of spreading a swarm of agents uniformly on a ring graph. Ant-like *oblivious* agents having limited capabilities are considered. The agents are assumed to have little memory, they all execute the same algorithm and no direct communication is allowed between them. Furthermore, the agents do not possess any global information. In particular, the size of the ring (n) and the number of agents in the swarm (k) are unknown to them. The agents are assumed to operate on an unweighted ring graph. Every agent can measure the distance to his two neighbors on the ring, up to a limited range of V edges.

The first task considered, is uniformly spread dynamical (i.e. in motion) deployment on the ring. We show that if either the ring is unoriented, or the visibility range is less than $\lfloor n/k \rfloor$, this is an impossible mission for the agents. Then, for an oriented ring and $V \geq \lceil n/k \rceil$, we propose an algorithm which achieves the deployment task within the optimal time. The second task discussed, called quiescent spread, requires the agents to spread uniformly over the ring and stop moving. We prove that under our model in which every agent can measure the distance only to his two neighbors, this task is impossible. Subsequently, we propose an algorithm which achieves quiescent and almost uniform spread.

The algorithms we present are scalable and robust. In case the environment (the size of the ring) or the number of agents changes during the run, the swarm adapts and re-deploys without requiring any outside interference.

1 Introduction

In this paper we consider multi-agent formation problems on a ring graph. The ring consists of n nodes and the number of agents will be denoted by k. Ant-like agents having limited capabilities are considered. The agents are assumed to be memoryless, all of them execute the same algorithm and no communication is allowed between them. The agents are *uniform* (or indistinguishable) in the sense that they have no identifiers hence can not be distinguished and they all follow the same algorithm. We also assume that the agents do not possess any global information e.g. the size of the ring (n) and the size of their swarm (k) are unknown to the agents. The agents operate on an unweighted ring graph. Every agent can measure the distance to his two neighbors on the ring up to a limited "visibility" range of V edges: if two adjacent agents are farther than V

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 215–226, 2010.

edges apart, they can not measure the distance between them. However, in this case, they know that the distance is larger than V.

We consider *oblivious* algorithms. An algorithm is *oblivious* if the action performed by an agent is dependent solely on the system state at the time the action is taken. In particular, each agent's current action is not dependent on past system states and on the agent's memory.

The algorithms we propose are scalable and robust, in the sense that if the environment (the size of the ring) or the number of agents change during the run, the swarm adapts and re-form without requiring any outside interference.

A question that naturally arises is the joint timing or synchronicity of the agents' operations in the environment. We use two synchronicity models by Suzuki *et al.*[9]. In the *semi-synchronous* model (SSM) the agents operate in cycles. In every cycle only a subset of the agents are active. It is guaranteed that in an infinite run, every agent will become active infinitely many times. In a time cycle, every active agent sense the environment and takes an action, either traversing an edge or staying in place. In this model, traversing an edge is assumed instantaneous. The *synchronous* model (SM) is similar to SSM but in every cycle all agents are active. It is important to distinguish SSM from CORDA[12]. In SSM (and SM) the look-compute-move cycle is atomic while in CORDA a finite (but unbounded) time may pass between sensing (the "look" phase) and acting (the "move" phase). Hence, in CORDA, the agents may take an action (move) according to an outdated information.

All the impossibility results presented in this paper are proved under SM. Since SM is a particular instance of SSM, the proofs hold for SSM as well. Furthermore, SM is also a particular instance of CORDA so our impossibility results hold for CORDA. All the algorithms we shall present work under both SM and SSM. Time bounds for SSM are presented in terms of *rounds* as defined below. Under SM, every time cycle is a *round*. So if algorithm A converges within t *rounds* under SSM then it converges within t time cycles under SM.

Definition 1. *A* round *is a time period in which every agent was active at least once.*

The tasks considered in this paper are two variants of the task of spreading a group of agents uniformly on a ring graph. Assuming the graph contains n vertices and the swarm is composed of $k \leq n$ agents, the most uniform spread possible is when the distance between any two adjacent agents is either $\lceil n/k \rceil$ or $\lfloor n/k \rfloor$.

Since the agents are indistinguishable and memoryless (i.e. are *oblivious*), a complete system description is given by the agents location. Formally, a system configuration is a vector of length k specifying the agents location.

Definition 2. *The agents form a balanced configuration if the distance between any two adjacent agents is either* $\lceil n/k \rceil$ *or* $\lfloor n/k \rfloor$.

The first task we consider is forming and maintaining balanced configurations i.e. the agents are required to form a balanced configuration and to stay in a balanced configuration. The agents may move and change configurations as long

as they maintain balanced configurations. We denote the task of forming and maintaining a balanced configuration by *uniform spread*.

We first show that in the case of unoriented ring i.e. the agents do not apriori agree on an orientation of the ring, no deterministic *oblivious* algorithm can consistently achieve *uniform spread*. Then, considering an oriented ring, we show that if the agent's visibility range (V) is strictly less than $\lfloor n/k \rfloor$, no *oblivious* algorithm can achieve *uniform spread*. Algorithm 1 presented in Section 4.2 is then prove to achieve *uniform spread* for $V \geq \lfloor n/k \rfloor$, therefore the analysis covers all possible values for V for the oriented ring. In case $V \geq \lceil n/k \rceil$, a swarm of agents running Algorithm 1 will achieve *uniform spread* within $O(n)$ time cycles under SM and $O(n)$ *rounds* under SSM. In case all agents start from the same vertex, *uniform spread* can not be achieved faster than $\Omega(n)$, hence the algorithm convergence time is optimal.

The second task we consider is quiescent *uniform spread* i.e. the agents are required to form a balanced configuration and stop moving. Stopping is of importance since moving consumes energy. We show that under our *oblivious* model in which every agent can sense only the distance to his two neighbors, forming a quiescent balanced configuration is impossible. Hence we introduce Algorithm 2 under which the agents form quiescent, but "semi-balanced" configurations.

Definition 3. *The agents form a semi-balanced configuration if the distance between any two adjacent agents is at least* $\frac{n}{k} - \frac{k}{2}$ *and at most* $\frac{n}{k} + \frac{k}{2}$.

In case $n/k \gg k$, a semi-balanced configuration is (almost) balanced. Algorithm 2 achieves quiescent semi-balanced configurations within $O(n)$ *rounds* (or time cycles). In case all agents start from the same vertex, a semi-balanced configuration can not be formed faster than $\Omega(n)$. Hence the convergence time of Algorithm 2 is optimal. We note that Algorithm 2 requires slightly larger visibility range, that is $V \geq \frac{n}{k} + \frac{k}{2}$.

Consider several agents on the same vertex. Since the agents are indistinguishable, the symmetry between them can not be broken and they can not be separated. We bypass this symmetry breaking problem by assuming distinct initial locations i.e. it is assumed that when the algorithm is initialized there are no two agents occupying the same vertex. Note that under the algorithms we present, if initially the agents were all at distinct vertices then there will never be two agents on the same vertex.

Due to space limitations, the proofs of the lemmas and some of the theorems are omitted here, they can be found in our readily available TR[6].

2 Related Work

If instead of a ring graph we consider the environment a continuous circle on the plane, we obtain a continuous space variant of our problem. The uniform circle formation problem is an instance of the general pattern formation problem, in which the agents are required to form a specific geometric constellation on the plane[9]. In one instance of the uniform circle formation problem, the agents are

required to uniformly spread over some circle on the plane, and the circle is not predetermined. The circle formation task may be divided into two sub-tasks: in the first sub-task the agents are required to form a non-uniform circle and, in the second, to spread uniformly over that circle.

Sugihara and Suzuki[13] suggested a heuristic algorithm for the limited visibility case. Later, Suzuki and Yamashita[14] proposed a non-*oblivious* algorithm in which the agents had the capability to remember past system states. Defago and Souissi[4] suggested the agreed upon circle to be the smallest circle enclosing all agents. Their algorithm requires full visibility, i.e. every agent sees all other agents and can perform global calculations. In their solution the agents converge toward uniform cycle and does not form it. Later, Chatzigiannakis *et al.*[2] simplified the algorithm proposed by Defago by modifying the model. Katreniak[10] solved the problem of forming a *bi-angular* circle. Dieudonne *et al.*[5] presented an algorithm which forms a uniform cycle starting from *bi-angular* circle for any number of agents but 4. Recently, Flocchini *et al.*[7] considered ϵ-*approximate* solutions. They proved that exact solution is impossible if the ring is not oriented. A different but related problem is spreading the agents uniformly over a line segment, see [15,11,3,1].

To the best of our knowledge, we are the first to consider the discrete case of the problem of balanced deployment on a ring environment (i.e. spreading over a ring graph) under limited visibility. Furthermore, in previous work regarding uniform spread over a continuous ring (or a line) the agent speed was unlimited i.e. the maximum step size an agent could take in a single time cycle was unbounded so the time bounds previously achieved were proportional only to the number of agents. In our work, we limit the agent speed to one edge per time cycle so the time bounds achieved are relative to the ring size as well.

3 Preliminaries

Denote the set of agents by A where $|A| = k$. Whenever we fix an agent a, let a_{+1} (a_{-1}) be the clockwise (counter-clockwise) neighbor of agent a on the ring graph. Similarly a_{+2} will denote the clockwise neighbor of a_{+1} and so on. A step taken by an agent clockwise is a "forward step" and a counter-clockwise step is a "backward step". Let $d(a, b)$ be the distance between agents a and b, defined as the number of edges on the clockwise path from a to b. Note that $d(a, b) + d(b, a) = n$. $d_{-1}(a)$ is the distance between a_{-1} and a; $d_{+1}(a)$ is the distance between a and a_{+1}. The set $A[a, b) \subseteq A$ includes, by definition, all the agents between a and b including a but not including b. Note that $A[a, b) + A[b, a) = A$. An illustration of these notations can be found in Figure 1.

Recall that V is the agents' visibility range. If $d(a, a_{+1}) \leq V$ then agents a and a_{+1} can measure $d(a, a_{+1})$. If $d(a, a_{+1}) > V$ the agents can not sense each other and will use $d(a, a_{+1}) = \infty$. We shall further assume that every agent can measure only the distance to his two neighbors an can not "see beyond them" e.g. even in the case where $d(a, a_{+2}) \leq V$, agent a can not measure the distance to a_{+2}.

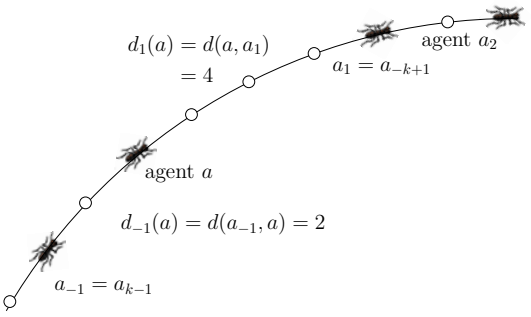


Fig. 1. Notations

Throughout the paper mathematical operations on agent indices are modulo k. When we explicitly add t to the indices of a quantity we refer to the value of that quantity at time t, e.g. $d\,(a, b; t)$ is the distance between agents a and b at time t.

4 Uniform Spread

4.1 Impossibility Results

For the case of unoriented rings i.e. when the agents have no access to a common orientation of the ring, *uniform spread* can not be achieved by a uniform deterministic algorithm. This holds even if the the agents have memory, global knowledge and unlimited sensing abilities. Like many other impossibility results in distributed systems, the swarm can not achieve *uniform spread* simply because the agents are unable to break symmetry. Our impossibility result is the discrete variant of a similar result by Flocchini *et al*[7]. The result we obtain is stronger since it holds under both SM and SSM where the continuous space result holds only for the SSM.

Theorem 1. *There is no uniform deterministic algorithm that achieves* uniform spread *on unoriented rings.*

Proof. We will prove the theorem under SM. Consider the case of an even number of agents k and a ring of size $n = m \cdot k$ where $m \geq 2$ is an integer. Fix an agent a, and consider a configuration $C\,(l)$ defined as follows: for every i such that $0 \leq i \leq k/2 - 1$,

$$d_{+1}\,(a_{2i}) = d_{-1}\,(a_{2i+1}) = 2l + 1$$
$$d_{-1}\,(a_{2i}) = d_{+1}\,(a_{2i-1}) = 2\,(m - l) - 1$$

where l is an integer such that $0 \leq l \leq m$. Note that configuration $C\,(l)$ is not balanced for any l.

Let A_{even} be the group of agents a_{+2i} where $0 \leq i \leq k/2-1$ and A_{odd} - the agents a_{+2i+1}. All agents of the set A_{even} sense the same world view. All agents of the set A_{odd} sense the same view. Furthermore, the view of the agents of A_{odd} is a "mirror image" of the view of the agents of A_{even} e.g. d_{+1} (resp. d_{-1}) of any agent of A_{even} equals d_{-1} (resp. d_{+1}) of any agent of A_{odd}. Because the algorithm is uniform and deterministic, when an agent of A_{even} takes a clockwise step, all other agents of A_{even} take a clockwise step and all agents of A_{odd} take a counter-clockwise step. The resulting configuration will again be a $C\,(l)$-type configuration (with a different l value) hence not balanced. This shows that there are always "initial" unbalanced class of configurations from which the agents applying a uniform deterministic algorithm will never escape.

In contrast to the result above, on oriented rings, *uniform spread* is possible. We shall also ask: "what is the minimal visibility range that enables *uniform spread*?". Theorem 2 below states that *uniform spread* is impossible if the agent's sensing range is strictly less than $\lfloor n/k \rfloor$. Algorithm 1, presented in section 4.2, converges to a balanced configuration for $V \geq \lfloor n/k \rfloor$ so the question is completely settled (the convergence proof for $V \geq \lfloor n/k \rfloor$ can be found in Appendix B of our TR[6]).

Theorem 2. *On an oriented ring, if the sensing range of the agents is strictly smaller than $\lfloor n/k \rfloor$, no uniform deterministic algorithm can achieve uniform spread.*

Proof. We shall again prove the theorem under SM. Consider $k \geq 3$ agents $a_{+1}...a_{+k}$ on a ring of size $n = m \cdot k - 1$ where $m \geq 2$ is an integer. Let the sensing range of the agents (V) be strictly smaller than $\lfloor n/k \rfloor$. Consider the non balanced configuration in which $d\,(a_{+1}, a_{+2}) = n - \lfloor n/k \rfloor\,(k-1) > \lceil n/k \rceil$, all other distances between adjacent agents being $\lfloor n/k \rfloor$. Here all the gaps are strictly greater than V so the agents can not sense each other. For every agent $d_{-1} = d_{+1} = \infty$. Since the algorithm is *oblivious,* all agents will take the same actions, hence forth the configuration will remain unchanged.

4.2 Uniform Spread on Oriented Ring

In order to bypass the impossibility result regarding unoriented rings we discuss oriented rings i.e. we assume that the agents agree on a common orientation of the ring (alternately, in order to break symmetry, one could consider probabilistic algorithms or use a different synchronicity model, see e.g. [8]). The *uniform spread* algorithm we propose is very simple. Every agent tries to balance the two distances to its nearest neighbor ahead (d_{+1}) and behind (d_{-1}). In order to break symmetry, we do not allow the agents to take forward steps. If $d_{-1} > d_{+1}$ the agent takes a backward step hence decreasing d_{-1} and increasing d_{+1}. If $d_{+1} > d_{-1}$ the agent will not move while other agents will act toward balancing the system.

The proposed algorithm is presented as Algorithm 1. A similar algorithm for the continuous space problem can be found in section 5.1 of [7]. The algorithm

Algorithm 1. Uniform Spread

```
if d_{-1} > d_{+1} then
    Take a step backward.
```

correctness and time complexity are discussed below. We show that the algorithm converges to a balanced configuration in $O(n)$ *rounds* if $V \geq \lceil n/k \rceil$. In case all agents start from the same vertex, a balanced configuration can not be achieved faster than $\Omega(n)$ so the algorithm convergence time is optimal.

The functions f, g are defined by

$$f(x) \triangleq x \cdot \lceil n/k \rceil$$
$$g(x) \triangleq x \cdot \lfloor n/k \rfloor$$

We show in Lemma 1 that $f(j-i)$ and $g(j-i)$ are respectively an upper and lower bound on the distance between any two agents a_{+i}, a_{+j} in a balanced configuration.

Lemma 1. *In a balanced configuration for every two agents* a_{+i}, a_{+j}

$$g(j-i) \leq d(a_{+i}, a_{+j}) \leq f(j-i)$$

For any two agents a_{+i}, a_{+j} let

$$u(a_{+i}, a_{+j}) \triangleq d(a_{+i}, a_{+j}) - f(j-i)$$
$$l(a_{+i}, a_{+j}) \triangleq g(j-i) - d(a_{+i}, a_{+j})$$

In case $u(a_{+i}, a_{+j}) > 0$, the agents a_{+i}, a_{+j} are too far apart. So the distance $d(a_{+i}, a_{+j})$ must be reduced by at least $u(a_{+i}, a_{+j})$ edges before the configuration is balanced. Since forward steps are not allowed, a_{+j} must take at least $u(a_{+i}, a_{+j})$ backward steps in order to reach a balanced configuration. Similarly, in case $l(a_{+i}, a_{+j}) > 0$, the agents are too close so the distance $d(a_{+i}, a_{+j})$ must be increased by a_{+i}.

Define the upper and lower loads on agent a by

$$u(a) \triangleq \max_j \{u(a_{+j}, a)\}$$
$$l(a) \triangleq \max_j \{l(a, a_{+j})\}$$

Intuitively, before forming a balanced configuration, agent a must take at least $\max\{u(a), l(a)\}$ steps. Next, two technical lemmas regarding the upper and lower loads are presented.

Lemma 2. *Let* a_{-i} *be the agent such that* $u(a_{-i}, a) = u(a)$ *and let* a_{+j} *be the agent such that* $l(a, a_{+j}) = l(a)$ *then*

1. $d_{+1}(a_{-i}) \geq \lceil n/k \rceil$.
2. $d_{-1}(a_{-i}) \leq d_{+1}(a_{-i})$.
3. $d(a_{j-1}, a_{+j}) \leq \lfloor n/k \rfloor$.
4. $d_{+1}(a_{+j}) \geq d_{-1}(a_{+j})$.

Lemma 3. *There is an agent a for which $u(a) = 0$ and there is an agent b for which $l(b) = 0$.*

For every agent a let $u_0(a)$ be the agent such that $u(u_0(a)) = 0$ and the cardinality of the set $A[a, u_0(a))$ is minimal. In case $u(a) = 0$ then $u_0(a) = a$ and $|A[a, u_0(a))| = 0$. Similarly, let $l_0(a)$ be the agent such that $l(l_0(a)) = 0$ and the cardinality of the set $A[a, l_0(a))$ is minimal. Such agents exist by Lemma 3. Define the upper and lower potentials of agent a by

$$v_u(a) \triangleq 2u(a) + |A[a, u_0(a))|$$
$$v_l(a) \triangleq 2l(a) + |A[a, l_0(a))|$$

The system upper and lower potentials are given by

$$V_u \triangleq \max_{a \in A} \{v_u(a)\}$$
$$V_l \triangleq \max_{a \in A} \{v_l(a)\}$$

By Lemma 1, the potential of a balanced configuration is zero. On the other hand, if $V_u = V_l = 0$ then the system is in a balanced configuration. We show in the next lemma that the upper and lower potentials of an agent do not increase.

Lemma 4. *Fix an agent a. Then $u(a)$, $v_u(a)$, $l(a)$ and $v_l(a)$ are non-increasing under SSM model.*

Recall that a *round* is a time period in which every agent was active at least once. The next two lemmas show that in every *round* both the upper and lower system potentials decrease. All the claims stated so far hold for any V. The next two lemmas hold for $V \geq \lceil n/k \rceil$ and $V \geq \lfloor n/k \rfloor$ respectively.

Lemma 5. *If $V_u(t_1) > 0$, $V \geq \lceil n/k \rceil$, and between times t_1 and t_2 all the agents were active at least once, $V_u(t_2) < V_u(t_1)$.*

Lemma 6. *If $V_l(t_1) > 0$, $V \geq \lfloor n/k \rfloor$, and between times t_1 and t_2 all the agents were active at least once, $V_l(t_2) < V_l(t_1)$.*

In order to bound the convergence time, the potential of the initial configuration is bounded in the next lemma. The proof is concluded in Theorem 4.

Lemma 7. *For any configuration, $V_u \leq 2n + k$ and $V_l \leq 2n + k$.*

Theorem 3. *A swarm of agents following Algorithm 1 with $V \geq \lceil n/k \rceil$ will form a balanced configuration within $2n + k$* rounds *under SSM and $2n + k$ time cycles under SM.*

5 Quiescent Uniform Spread

5.1 Impossibility Result

The next theorem proves that achieving quiescent balanced configuration is impossible under our *oblivious* model in which every agent can sense only the distance to his two neighbors. The result holds for any V. The impossibility result does not hold under a stronger model in which every agent can sense all agents in his visibility range. This stronger model is beyond the scope of this paper and will be analyzed elsewhere.

Theorem 4. *If every agent can sense the distance to only his two neighbors on the ring, there is no* oblivious *deterministic algorithm that achieves* quiescent uniform spread.

Proof. Assume toward contradiction that a swarm of agents following an *oblivious* algorithm A form a quiescent balanced configuration. Fix the swarm size (k) and consider a ring of n vertices where n/k is an integer. There is a single balanced configuration in which for every agent $d_{+1} = d_{-1} = n/k$. So in this configuration, all agents are quiescent. Consider a ring of n vertices where n/k is a fraction. In every balanced configuration there is an agent a such that $d_{+1}(a) = \lceil n/k \rceil$, $d_{-1}(a) = \lfloor n/k \rfloor$ and there is an agent b such that $d_{+1}(b) = \lfloor n/k \rfloor$, $d_{-1}(b) = \lceil n/k \rceil$. Since one of the balanced configuration must be quiescent, agents a and b do not move. Algorithm A is *oblivious* so all agents run the same algorithm. We conclude that every agent such that $|d_{+1} - d_{-1}| \leq 1$ does not move under algorithm A.

To construct the counter example fix $k \geq 4$ and let $n = m \cdot k + 4$ where m is a positive integer. Assume the following initial configuration:

$$\begin{aligned} d(a_{+1}, a_{+2}) &= m + 1 \\ d(a_{+2}, a_{+3}) &= m + 2 \\ d(a_{+3}, a_{+4}) &= m + 1 \end{aligned}$$

and all other distances equal m. For every agent $|d_{+1} - d_{-1}| \leq 1$ so all agents do not move and the configuration remains unchanged indefinitely. However, the configuration is not balanced, a contradiction.

5.2 Quiescent Semi-stable Configuration

In the previous section we have shown that forming a quiescent balanced configuration is impossible. The impossibility result is based on the observation that under any algorithm, every agent for which $|d_{+1} - d_{-1}| \leq 1$ must not move. Based on that observation we present Algorithm 2 in which every agent is quiescent as long as

$$d_{-1} - d_{+1} \leq 1 \tag{1}$$

Algorithm 2. Quiescent Stable Configuration

```
if d_{-1} > d_{+1} + 1 then
    Take a step backward.
```

We show in Lemma 8 that when Equation 1 holds for all agents, they form a semi-balanced configuration (recall Definition 3 from section 1). The algorithm requires a slightly larger visibility range i.e. $V \geq \frac{n}{k} + \frac{k}{2}$. Note that in case $n/k \gg k$, a semi-balanced configuration is (almost) balanced. The algorithm correctness and time complexity are discussed below.

Lemma 8. *If Equation 1 holds for all agents then the agents form a semi-balanced configuration.*

The convergence proof of Algorithm 2 resembles the proof of Algorithm 1. The proof is based on potential functions which are similar to the ones used in the proof of Algorithm 1.

The function $h(i,d)$ is defined by

$$h(i,d) \triangleq i \cdot d + \sum_{j=1}^{i} j = i \cdot d + \frac{i(i+1)}{2}$$

For every agent a let

$$s(a_{-i},a) \triangleq d(a_{-i},a) - h(i, d_{+1}(a))$$
$$s(a) \triangleq \max_i \{s(a_{-i},a)\}$$

The intuition behind the definition of s is the following: if $s(a) \leq 0$ then Equation 1 holds for agent a. Therefore, by Lemma 8, if for all agents $s = 0$ then they form a semi-balanced configuration. Define the potentials of an agent and of a system respectively via:

$$v_s(a) \triangleq \begin{cases} 2 \cdot s(a) + d_{+1}(a) & s(a) > 0 \\ 0 & else \end{cases}$$
$$V_s \triangleq \max_{a \in A} \{v_s(a)\}$$

When $V_s = 0$, the agents form a semi-balanced configuration. The next few lemmas and theorem show that V_s will be zeroed within $3n$ *rounds*. We first present two technical lemmas regarding s and v_s.

Lemma 9. *Let a_{-i} be an agent such that $s(a) = s(a_{-i},a)$ then*

1. $d_{+1}(a_{-i}) \geq h(i, d_{+1}(a)) - h(i-1, d_{+1}(a))$.
2. $d_{-1}(a_{-i}) \leq d_{+1}(a_{-i}) + 1$.

Lemma 10. *Let a be an agent such that $V(s) = v_s(a) > 0$ then,*

1. $d_{-1}(a) \geq d_{+1}(a) + 2$.
2. $d_{-1}(a_{+1}) \leq d_{+1}(a_{+1}) + 1$.
3. $d_{+1}(a) \leq \frac{n}{k} + \frac{k}{2}$.

The next two lemmas show that V_s decreases with time. To be exact, every *round*, the system potential decreases by at least one.

Lemma 11. *If $v_s(a;t) < V_s(t)$ then $v_s(a;t+1) < V_s(t)$ under SSM.*

Lemma 12. *If $V_s(t_1) > 0$, $V \geq \frac{n}{k} + \frac{k}{2}$, and between times t_1 and t_2 all the agents were active at least once, $V_s(t_2) < V_s(t_1)$.*

The next lemma and theorem complete the proof in the following manner: the potential of the initial configuration is bounded by $3n$ and in every *round* the potential decreases. Therefore within $3n$ rounds $V_s \leq 0$ and the agents reach a quiescent semi-balanced configuration.

Lemma 13. *For any configuration $V_s \leq 3n$.*

Theorem 5. *A group of k agents following Algorithm 2 where $V \geq \frac{n}{k} + \frac{k}{2}$ will form a quiescent semi-balanced configuration within* $3n$ rounds *under SSM and $3n$ time cycles under SM.*

6 Conclusion

In this paper we have considered two variants of the problem of spreading a swarm of agents uniformly on a ring graph. In the first variant the agents are required to "dynamically" spread uniformly over the ring. We have shown that if the ring is unoriented or the sensing range of the agents is strictly less than $\lfloor n/k \rfloor$, this task is impossible. Considering an oriented ring and $V \geq \lceil n/k \rceil$, we have proposed an algorithm which achieves the task within optimal time. In the second variant, the agents are required to spread over the ring and stop once a balanced configuration is reached (quiescent spread). We have shown that under our model in which every agent can measure the distance only to his two neighbors, achieving this task is impossible. As a partial solution, we have proposed an algorithm which achieves quiescent and "almost uniform" spread.

Some interesting open issues that remain to be answered are:

- In continuous space systems, the agent's speed is limited. Can our algorithms induce *ϵ-approximate* [7] algorithms for the limited speed continuous space case? If so, the time bounds achieved shall be with respect to the ring size, agent speed and the number of agents as opposed to state of the art bounds which consider the number of agents only.
- In case every agent can measure the distance to all agents in his visibility range, what is the minimal visibility range that enables a quiescent balanced configuration? Which algorithms achieve that?
- Under our model in which every agent can sense the distance to his two neighbors only, can the agents achieve a more uniform quiescent spread than the semi-balanced configuration we have presented? Perhaps a random algorithm can achieve better results on average?

Acknowledgments. This research was supported by the Technion Goldstein UAV and Satellite Center and by the European Community's FP7-FET program, SMALL project.

References

1. Carli, R., Bullo, F.: Quantized coordination algorithms for rendezvous and deployment. Sicon (2009)
2. Chatzigiannakis, I., Markou, M., Nikoletseas, S.E.: Distributed circle formation for anonymous oblivious robots. In: Ribeiro, C.C., Martins, S.L. (eds.) WEA 2004. LNCS, vol. 3059, pp. 159–174. Springer, Heidelberg (2004)
3. Cohen, R., Peleg, D.: Local spreading algorithms for autonomous robot systems. SIROCCO 2006 399(1-2), 71–82 (2008); Structural Information and Communication Complexity (SIROCCO 2006)
4. Defago, X., Souissi, S.: Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. Theoretical Computer Science 396(1-3), 97–112 (2008)
5. Dieudonne, Y., Labbani-Igbida, O., Petit, F.: Circle formation of weak mobile robots. ACM Trans. Auton. Adapt. Syst. 3(4), 1–20 (2008)
6. Elor, Y., Bruckstein, A.M.: Multi-agent deployment and patrolling on a ring graph. Tech. rep., Computer Science Department. Technion Haifa, Israel (September 2009), http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-info.cgi/2009/CIS/CIS-2009-16
7. Flocchini, P., Prencipe, G., Santoro, N.: Self-deployment of mobile sensors on a ring. Theor. Comput. Sci. 402, 67–80 (2008)
8. Gordon, N., Elor, Y., Bruckstein, A.M.: Gathering multiple robotic agents with crude distance sensing capabilities. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) ANTS 2008. LNCS, vol. 5217, pp. 72–83. Springer, Heidelberg (2008)
9. Hideki, Y., Oasa, A., Suzuki, I.: Formation and agreement problems for synchronous mobile robots with limited visibility. In: Proceedings of the 1995 IEEE International Symposium on Intelligent Control, pp. 453–460 (August 1995)
10. Katreniak, B.: Biangular Circle Formation by Asynchronous Mobile Robots. In: Pelc, A., Raynal, M. (eds.) SIROCCO 2005. LNCS, vol. 3499, pp. 185–199. Springer, Heidelberg (2005)
11. Martinez, S., Bullo, F., Corts, J., Frazzoli, E.: On synchronous robotic networks-part ii: Time complexity of rendezvous and deployment algorithms. IEEE Transactions on Automatic Control 52(12), 2214–2226 (2007)
12. Prencipe, G.: Instantaneous actions vs. full asynchronicity: Controlling and coordinating a set of autonomous mobile robots. In: Restivo, A., Ronchi Della Rocca, S., Roversi, L. (eds.) ICTCS 2001. LNCS, vol. 2202, pp. 154–171. Springer, Heidelberg (2001)
13. Sugihara, K., Suzuki, I.: Distributed algorithms for formation of geometric patterns with many mobile robots. Journal of Robotic Systems 13(3), 127–139 (1996)
14. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM Journal on Computing 28(4), 1347–1363 (1999)
15. Wagner, I.A., Bruckstein, A.M.: Row straightening via local interactions. Circuits, Systems, and Signal Processing 16(3), 287–305 (1997)

Multi-Swarm Optimization for Dynamic Combinatorial Problems: A Case Study on Dynamic Vehicle Routing Problem

Mostepha Redouane Khouadjia[1], Enrique Alba[2], Laetitia Jourdan[1], and El-Ghazali Talbi[1]

[1] National Institute for Research in Computer Science and Control (INRIA) Lille, France
[2] Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, E.T.S. Ingeniería Informática, Campus de Teatinos, Málaga, Spain
mostepha-redouane.khouadjia@inria.fr, eat@lcc.uma.es, laetitia.jourdan@inria.fr, talbi@lifl.fr

Abstract. Many combinatorial real-world problems are mostly dynamic. They are dynamic in the sense that the global optimum location and its value change over the time, in contrast to static problems. The task of the optimization algorithm is to track this shifting optimum. Particle Swarm Optimization (PSO) has been previously used to solve continuous dynamic optimization problems, whereas only a few works have been proposed for combinatorial ones. One of the most interesting dynamic problems is the Dynamic Vehicle Routing Problem (DVRP). This paper presents a Multi-Adaptive Particle Swarm Optimization (MAPSO) for solving the Vehicle Routing Problem with Dynamic Requests (VRPDR). In this approach the population of particles is split into a set of interacting swarms. Such a multi-swarm helps to maintain population diversity and good tracking is achieved. The effectiveness of this approach is tested on a well-known set of benchmarks, and compared to other metaheuristics from literature. The experimental results show that our multi-swarm optimizer significantly outperforms single solution and population based metaheuristics on this problem.

1 Introduction

While most of the optimization problems discussed in the scientific literature are static, many real-world problems change over time, i.e. they are dynamic. In those cases, the optimization algorithm has to track a moving optimum as closely as possible. In dynamic continuous problems, it has been argued that the Particle Swarm Optimization (PSO) is potentially well-suited to track the shifting optimum [6,14,5]. PSO needs to be adapted for optimal results on dynamic optimization problems. In this paper we propose two adaptation levels. The first level is given by the Adaptive Particle Swarm Optimization (APSO) algorithm which provides an adaptive approach based on explicit memory. This

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 227–238, 2010.

memory stores the gathered information during the previous searches. The second level of adaptation is given by the Multi-swarm APSO (MAPSO) approach through better population diversity. Thanks to recent advances in information and communication technologies (geographic information systems (GIS), global positioning systems, (GPS), traffic flow sensors,...), which are able to reduce costs and improve transport services, we can now timely generate routes as soon as new event occurs. Thus, Dynamic Vehicle Routing Problem (DVRP) is getting increasing importance this last decade. The dynamism is expressed in this problem by the changes that happen in a repeated manner (at each new customer demand). We can suggest that the DVRP landscape is similar to a multidimensional landscape consisting of several peaks (optima), where the height, the width and the position of each peak is altered every time a change in the environment occurs. So, the multi-swarm approach is justified in this kind of fitness landscape.

We present in this paper a multi-swarm approach for solving the Vehicle Routing Problem with Dynamic Requests (VRPDR). An effective representation for particles and for this problem is provided. To the best of our knowledge, this is the first PSO implementation for this particular version of DVRP. It is tested using data sets introduced in [13], and compared to other well-known metaheuristics. The remainder of this paper is organized as follows: Section 2 discusses several multi-population approaches for dynamic optimization problems. In Section 3 a formal presentation of the problem is given. Section 4 describes the proposed MAPSO approach. Section 5 provides experimental analysis. Conclusion and future work are given in Section 6.

2 Multi-population Approaches for Dynamic Optimization Problems

Different multi-population approaches exist for dynamic environments. In the area of Evolutionary Algorithms (EAs), Oppacher and Wineberg in [17] propose a Shifting Balance Genetic Algorithm (SBGA) that consists in dividing the EA population into one main population and a number of smaller colony subpopulations. The task of the main population is to exploit the best optimum found, while the colony populations are forced to explore the different areas of the fitness landscape. A repulsion mechanism is introduced whenever a colony population gets too close to the core population thus driving the colonies far from the core population. Periodically, the colonies update the core population by sending some emigrant solutions. One of the most promising approaches over the last decade is the Self-Organizing Scouts (SOS) proposed by Branke [8]. Opposite to SBGA, the goal here is to have a number of subpopulations (scouts) watching over the best local optima, and one single population which explores the fitness landscape. When a local optimum is discovered, a part of the population is split off and remains close to this optimum for further exploration. The remainder of the population continues the search for new local optima that can appear when the environment changes. Also, Ursem proposes in [22] the

Multinational Genetic Algorithm (MGA) which structures the population into subpopulations using a procedure called *hill-valley detection*. For two points in the search space, a random sample on the line between these two end points is evaluated. The valley is detected if the fitness of the sample is lower than the fitness of the two end points. This method is used to determine if an individual is not located on the same peak with the rest of its population, and hence it should migrate to a different population. The procedure can also lead to the merging of two populations if it finds that they are situated on the same peak.

The Multi-population approach was used also to enhance metaheuristics for diversity. One of the most popular algorithm used in this sense is the Particle Swarm Optimization (PSO). A Multi-population Charged PSO (MCPSO) is proposed in [4], inspired by atomic analogy. Repulsion mechanism is used between charged and neutral subpopulations. Its aim is to avoid the convergence of the population around the same local optima. The idea is to place a swarm on each local optimum in a multi-modal environment. Whilst the neutral subswarms continue to optimize, the surrounding charged particles maintain enough diversity to track dynamic changes in location of the covered peaks. This approach was extended by the authors to quantum model then introducing Multi-Quantum Swarm (MQS). The multi-swarm model has also been analysed by Parrott and Li [18]. There, the number and the size of swarms are adjusted dynamically by a speciation and crowding mechanisms for finding several optima in multimodal landscapes, also preventing overcrowding at peaks. In [5], Blackwell et al. reused the MQS with exclusion between swarms, and an additional anti-convergence mechanism which reinitializes the worst swarm once it has converged. Inspired by multi-swarm approach, we investigate in Section 4 whether a multi-population metaheuristic might also be beneficial in dynamic vehicle routing environments. The aim is to place different swarms on the search space to counterbalance the diversity loss of population.

3 Problem Description

A general description of the Vehicle Routing Problem is first given in order to introduce the Vehicle Routing Problem with Dynamic Requests (VRPDR).

3.1 The Static Vehicle Routing Problem

The static Vehicle Routing Problem (VRP) [10] is a well-known NP-hard combinatorial optimization problem which is encountered frequently in distribution logistics and transportation systems. It consists in designing routes for a fleet of capacitated vehicles that are to service a set of geographically dispersed points (customers, stores, cities,...) at the least cost (distance, time, number of vehicles,...). The VRP can be modelled mathematically using an undirected graph $G = (C, E)$, where $C = \{c_0, c_1, ..., c_n\}$ is the vertex set, and $E = \{(c_i, c_j) | c_i, c_j \in C, i < j\}$ is the edge set. A set of m homogeneous vehicles, each with capacity Q, originate from a single depot represented by the vertex

c_0. The vehicles must service all the customers represented by the set C. The quantity of goods q_i requested by each customer c_i is associated with the corresponding vertex. The goal is to find a feasible set of routes with the minimum total traveled distance. The solution is said feasible if each node is visited exactly once (i.e, it is included into exactly one tour), each tour starts and ends at the depot (c_0), and the sum of the quantities associated with the vertices contained in it never exceeds the corresponding vehicle capacity Q. More formally, the problem can be described as follows:

$$\textbf{Minimize} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \sum_{k=1}^{m} x_{ij}^{k} \tag{1}$$

Where:
n: number of vertices,
m: number of vehicles,
d_{ij}: cost or distance between the vertex c_i and the vertex c_j,
and

$$x_{ij}^{k} = \begin{cases} 1, \text{ if } (c_i, c_j) \text{ is covered by the vehicle } k \\ 0, \textit{ otherwise} \end{cases} \tag{2}$$

3.2 The Dynamic Vehicle Routing Problem

The Dynamic Vehicle Routing Problem (DVRP) [19] is strongly related to the static VRP. It can be described as a VRP in which information about the problem can change during the optimization process, contrary to the static VRP in which customers are known prior to the planning routes. The changing information involves several factors such as traveling time, traffic incidents, arrival of new customer demands, etc. In the Vehicle Routing Problem with Dynamic Requests (VRPDR), customer orders are revealed incrementally over time, although some orders may be known in advance at the beginning of the route design. The new requests lead to the reconfiguration of the current established plan in order to include these requests. We investigate here, the model introduced by Kilby et al. [13] and further refined by Montemanni et al. [16]. The proposed model makes a partition of the working day into time slices, and solves a partial problem at each time slice. In [13], Kilby adapts classical VRP benchmarks for this dynamic context. He studies the effect of the degree of dynamism on the solution quality.The proposed algorithm for the construction of routes is the basic insert and improvement algorithm. Besides, Montemanni et al. [16] propose a solving strategy based on the Ant Colony System paradigm. Meanwhile, Hanshar et al. [11] solve the problem with Genetic Algorithms and Tabu Search. A basic example of a VRPDR scenario is shown in Figure 1. In the example, two uncapacitated vehicles must service both advance (known) and dynamic request customers. Advance requests are represented with white nodes, while those that are dynamic are depicted by black nodes. The solid lines represent the two routes that the dispatcher has planned before the vehicles leave the depot, while the new route segments are indicated by dotted lines.

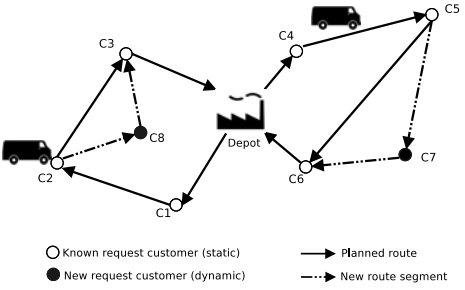


Fig. 1. Illustrating the Dynamic Vehicle Routing Problem

4 Multi-Adaptive Particle Swarm Optimization

In this section we present our Multi-Adaptive Particle Swarm Optimization (MAPSO) beginning by the canonical PSO and followed by the APSO approach and the dedicated APSO-DVRP. At the end, a multi-swarm model for APSO is given.

4.1 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithm was proposed by Kennedy and Eberhart [12]. It is a population based technique inspired by models of social swarm and flock behavior. A particle is defined by its position $\overrightarrow{x_i}$, the position of its personal best solution found so far $\overrightarrow{p_i}$, and its velocity $\overrightarrow{v_i}$. Furthermore, each particle knows the best position found so far by the global swarm $\overrightarrow{p_g}$. The algorithm proceeds iteratively, updating first all velocities, and then all particle positions as follows:

$$\overrightarrow{v}_i = \omega \overrightarrow{v}_i + \varphi_1 \times r_1(\overrightarrow{p}_i - \overrightarrow{x}_i) + \varphi_2 \times r_2(\overrightarrow{p}_g - \overrightarrow{x}_i) \tag{3}$$

$$\overrightarrow{x}_i = \overrightarrow{x}_i + \overrightarrow{v}_i \tag{4}$$

The Equation (3) is used to calculate the i-th particle's new velocity by taking into account three terms: the particle's previous velocity, the distance between the particle's best and current position, and finally, the distance between swarm's best position and the i-th particle's current position. Then, the i-th particle moves towards a new position according to the Equation (4). The role of the inertia weight ω is to regulate the moving velocity of the particles. The parameters φ_1, φ_2 control the relative attraction of the personal best and global best found solutions. Finally, r_1, r_2 are random variables drawn with uniform probability from $[0, 1]$.

4.2 Adaptive Particle Swarm Optimization

Some papers produced in the area of dynamic optimization deal with the non-stationary feature by regarding each change as the arrival of a new optimization

problem that has to be solved from scratch [3]. However this simple approach is impractical in some problems like DVRP, because solving a problem from scratch without reusing information from the past is meaningless, due to the fact that vehicles are already on routes and are servicing customers. Better solutions could be achieved by using an optimization algorithm that is capable of reusing the information gained in the past. Many authors have addressed the issue of transferring information from the old environment to the new one by enhancing the algorithms [7,23]. Usually, the approaches are enhanced with some sort of memory that might allow to store good solutions and reuse them later as necessary. This memory may be provided implicitly by using redundant representations, or explicitly by introducing an extra memory and formulating strategies to store in and retrieve solutions from that memory. The latter alternative seems to be more promising. In PSO this explicit memory is intrinsic to the algorithm, since each particle is defined by its current position and its best position so far. In order to have a better response to environment changes, we propose an Adaptive Particle Swarm Optimization (APSO). The adaptability of the algorithm consists in using the information gathered previously on the search space. When the environment changes, this mechanism allows the algorithm to restart the search from the best solutions found during the previous searches. Several memory schemes can be used [23]. The policy used in our approach consists at each change of environment, instead of blindly replacing all the $\overrightarrow{p_i}$ by their corresponding $\overrightarrow{x_i}$ at the initialization step, we found that it was worthwile to see the fitness of the $\overrightarrow{p_i}$. If this latter is better than the current position of the particle then we replace the current position by the best one. This method allows to give better response to the dynamic environment.

4.3 Adaptive Particle Swarm for Solving Dynamic Vehicle Routing Problem

Particle Representation: Usually in the literature, authors propose an indirect real number coding for the particles as in [1]. Each dimension should be rounded to the closest integer number and sorted. It operates with difficulty and consumes much CPU time during the decoding step. Meanwhile, if the position presents an infeasible solution, it is very difficult to repair.

In our approach, we propose a simple discrete representation which expresses the route of vehicles over the n customers to serve. We set up a dedicated solution encoding for the DVRP problem. The representation allows the insertion of dynamic customers in the already planned routes. Since the customer requests arrive along the time, it is necessary to have some kind of knowledge about the state of each customer (served/not served) and his processing time (i.e., the time in which he is served). On the other side, we keep some information about each vehicle. This information is related to its current position in the service area, its remaining capacity, its traveled distance, and its status (committed/not committed). The representation of each route R is a permutation of n customers, starting and ending at the depot ($c_0 = c_{n+1}$) as follows:

$$R : (c_0, c_1, c_2, ..., c_i, ..., c_n, c_{n+1}) \tag{5}$$

For each customer c_i, we assign the following information:

- (x_i, y_i): Coordinates of the the customer c_i.
- s_i: Boolean variable which indicates if the customer c_i has been already served or not.
- t_i: Processing time of the customer c_i.

Furthermore, for each route R served by the vehicle v_j we keep these information:

- (x_j, y_j): Coordinates of the vehicle v_j.
- cap_j: Remaining capacity of the vehicle v_j.
- $dist_j$: Distance traveled by the vehicle v_j.
- $commit_j$: Boolean variable which indicates if the vehicle v_j is committed to serve customers or not.

The initial population of APSO is obtained by generating a random permutation of customers who were left over from the previous working day. At each time slice, the new customers are inserted into the routes. The velocity vector $\overrightarrow{v}_i$ of the particle is initialized for each new dimension by a random number between $[1, m]$, where m is the number of the planned vehicle routes. The normalization of particle's velocity in index sequence allows to have feasible solutions. For each dimension of position vector $\overrightarrow{x_i}$, if the customer is served, we can not change neither the tour nor the position of this latter in the already established vehicle routes. The updating process is very similar to the *ejection chain* method that has been applied successfully to vehicle routing [20]. It consists in generating a compound sequence of interrelated simple moves between routes, leading from one solution to another. The customers move from their route and inserted into another one according to the cheapest cost insertion (i.e., the position which minimizes the cost of the insertion into the route).

Hybridizing APSO-DVRP: The hybridization scheme used here consists in a low-level teamwork hybrid (LTH). In [21], Talbi describes this class of hybrids as algorithms in which a given metaheuristic is embedded into another one. We have used the 2-Opt [15] heuristic as a local search for APSO. The 2-Opt heuristic is applied in APSO after the move of particles. A naive exploration of the neighborhood of a solution s is a complete evaluation of the objective function for every candidate neighbor s' of $N(s)$. For more efficiency, our local search is designed in a manner in which we can avoid the whole route evaluation. A way to evaluate the set of candidates is the evaluation $\triangle(s, m)$ of the objective function, where s is the current solution and m is the applied move. This incremental evaluation consists in evaluating only the transformation $\triangle(s, m)$ applied to a solution s rather than the complete evaluation of the neighbor solution $f(s') = f(s \oplus m)$. This is an important issue in terms of efficiency and must be taken into account in the design of high-achieving metaheuristics especially in dynamic optimization context [21].

4.4 Multi-Swarm Optimizer

Due to diversity loss and linear collapse [6], the swarm can converge quickly. The particles will be close to the global best attractor and the swarm will be shrinking at a rate determined by the inertia weight and by the local environment at the optimum. If the optimum shifts within the collapsing swarm, then re-optimization will be efficient. However, if the optimum shift is significantly far from the swarm, the low velocities of the particles will inhibit tracking, and the swarm can even oscillate around a false attractor and far from the true optimum (linear collapse). One of the approaches to delay the effect of diversity loss is a multi-swarm, with the aim of maintaining a multitude of swarms on different peaks [4]. We propose here a Multi-population version of APSO (MAPSO). The underlying idea is to place an APSO swarm on each local optimum of the DVRP fitness landscape. These swarms will maintain hopefully enough diversity to track dynamic changes in location of the covered peaks, and exchange information about each peak when change occurs in environment. To track the optimum in such an environment, the algorithm has to be able to follow a moving peak, and to jump to another peak when the peak heights change in a way that makes a previously non-optimal peak the lowest peak (minimization). In the DVRP fitness landscape, the lowest peak is the solution of feasible set of routes with the minimum total traveled distance. In designing a parallel cooperative model for any metaheuristic, the same questions need to be answered [2,21]:

- **The exchange decision criterion:** The exchange of information between the swarms is decided in a periodic way. In this work, a periodic exchange occurs in each algorithm after a fixed number of evaluations.
- **The exchange topology:** The communication exchange topology indicates for each APSO swarm its neighbor(s) regarding the exchange of information, that is, the source/destination algorithm(s) of the information. The ring topology is used in our approach.
- **The exchanged information:** This parameter specifies the information to be exchanged between the metaheuristics. It contains elite particles (i.e., with best personal positions) that have been previously found. The number of solutions to exchange is given by a percentage of the population.
- **The integration policy:** The integration policy deals with the usage of the received information. For our algorithm, an elitist replacement strategy is applied to integrate the received k particles by replacing the k worst particles of the local swarm if the incoming ones are better than them. Since the heterogeneous nature of the parallel environment (communication latency, performance of nodes), at a given moment of the optimization process, the subpopulations can be in different steps of the problem evolution. The migration process may involve a situation in which the immigrants could be found in host population that evolves either ahead or behind their base population. Each particle is labelled before its sending by the current time step (evolution step) of the problem on which it deals. Three cases may arise during the migration of particles towards a host population. The policy adopted for each case is as follows:

- If the immigrant particle is coming from a population working on the same partial problem as the host population: integrate the particle without any measure into the local population.
- If the immigrant particle is coming from a population working on a partial problem which is advanced in time than the host population: keep the particle in a waiting queue until the local problem reaches the evolution step of particle's problem, and integrate this latter into the population.
- If the immigrant particle is coming from a population working on a partial problem which is behind the host population: evolve the particle to the current evolution state of the local partial problem before integrating it into the population.

5 Experimental Analysis

The results achieved by our algorithm are presented in this section. The MAPSO algorithm has been implemented on ParadisEO framework [1] dedicated to design of metaheuristics, and the experiments were carried out using the Grid'5000[2] experimental testbed. In order to compare our algorithms with the reported results of other approaches on DVRP [11], a number of parameters need to be set in the benchmarks. To standardize the benchmarks[3], Montemanni et al. [16] fixed some benchmark parameters that can affect the final travel distances. The first is n_{ts}, the number of time slices in the optimization process. This parameter subdivided the day T into discrete time periods T_s, in which optimization is carried out on each one. Montemanni et al. [16] found that setting the parameter to $n_{ts} = 25$ yielded the best tradeoff between the objective value and computational cost. Secondly, the cutoff time T_{co} was set to 0.5. This means that the orders which arrive after the half of the working day ($0.5 \times T$) are postponed to the following day. Finally, the advanced commitment time T_{ac} was set to 0. This parameter has been considered to give the drivers an appropriate reaction time after having been committed to the new orders. The algorithm parameters used for our approach are summarised in the Table 1. For each instance, 30 independent runs of our algorithm have been considered. The choice to set the stopping criterion of the algorithm as the number of evaluations carried out on the population allows to standardize the comparison protocol between different metaheuristics. However, the other works use CPU time [16,11]. This latter is highly dependent on hardware, and not a suitable stop condition for this comparison.

5.1 Numerical Results

A comparison of the solutions quality in term of minimizing travel distances is done between our hybrid MAPSO_{2-Opt}, and several algorithms proposed previously in literature as: Ant System (AS) [16], Genetic Algorithm (GA_{2-Opt}),

[1] http://paradiseo.gforge.inria.fr
[2] https://www.grid5000.fr
[3] http://www.fernuni-hagen.de/WINF/inhalte/benchmark_data.htm

Table 1. Algorithm parameters for MAPSO metaheuristic

Parameter type	Default value	Range
Number of swarms	8	–
Population size per swarm	100	–
Migration topology	directional ring	–
Migration frequency	1000 evaluations	–
Migration size	5% population size	–
Inertia weight (ω)	1	–
φ_1	0.75	0.5-1.0
φ_2	0.5	0.5-1.0
Stopping criterion	5000 evals per Ts	25×5000=125000 evals per T

Table 2. Numerical results obtained by MAPSO_{2-Opt} compared to AS, and the hybrids GA_{2-Opt} and TS_{2-Opt}

Instances	Metaheuristics							
	MAPSO_{2-Opt}		AS [16]		GA_{2-Opt} [11]		TS_{2-Opt} [11]	
	Best	**Average**	**Best**	**Average**	**Best**	**Average**	**Best**	**Average**
c50	571.34	610.67	631.30	681.86	**570.89**	***593.42***	603.57	627.90
c75	**931.59**	***965.53***	1009.36	1042.39	981.57	1013.45	981.51	1013.82
c100	**953.79**	***973.01***	973.26	1066.16	961.10	987.59	997.15	1047.60
c100b	**866.42**	***882.39***	944.23	1023.60	881.92	900.94	891.42	932.14
c120	**1223.49**	***1295.79***	1416.45	1525.15	1303.59	1390.58	1331.22	1468.12
c150	**1300.43**	***1357.71***	1345.73	1455.50	1348.88	1386.93	1318.22	1401.06
c199	**1595.97**	***1646.37***	1771.04	1844.82	1654.51	1758.51	1750.09	1783.43
f71	287.51	***296.76***	311.18	358.69	301.79	309.94	**280.23**	306.33
f134	15150.5	16193	**15135.51**	16083.56	15528.81	***15986.84***	15717.90	16582.04
tai75a	1794.38	***1849.37***	1843.08	1945.20	1782.91	1856.66	**1778.52**	1883.47
tai75b	**1396.42**	***1426.67***	1535.43	1704.06	1464.56	1527.77	1461.37	1587.72
tai75c	1483.1	1518.65	1574.98	1653.58	1440.54	***1501.91***	**1406.27**	1527.72
tai75d	**1391.99**	***1413.83***	1472.35	1529.00	1399.83	1422.27	1430.83	1453.56
tai100a	**2178.86**	***2214.61***	2375.92	2428.38	2232.71	2295.61	2208.85	2310.37
tai100b	**2140.57**	2218.58	2283.97	2347.90	2147.70	***2215.93***	2219.28	2330.52
tai100c	**1490.4**	***1550.63***	1562.30	1655.91	1541.28	1622.66	1515.10	1604.18
tai100d	1838.75	1928.69	2008.13	2060.72	**1834.60**	***1912.43***	1881.91	2026.76
tai150a	**3273.24**	***3389.97***	3644.78	3840.18	3328.85	3501.83	3488.02	3598.69
tai150b	**2861.91**	***2956.84***	3166.88	3327.47	2933.40	3115.39	3109.23	3215.32
tai150c	**2512.01**	***2671.35***	2811.48	3016.14	2612.68	2743.55	2666.28	2913.67
tai150d	**2861.46**	***2989.24***	3058.87	3203.75	2950.61	3045.16	2950.83	3111.43
Total	**48104.13**	***50349.66***	50876.23	53794.02	49202.73	51089.37	49987.8	52725.85

and Tabu Search (TS_{2-Opt}), both proposed in [11]. Table 2 gives the obtained results of the algorithms and they are compared. The best and the average distances of the different algorithms are reported. Bolded entries indicate where the best solutions were obtained. We can see that the hybrid MAPSO is able to provide the higher quality solutions. MAPSO_{2-Opt} outperforms the rest and gives 15 new (unseen) best results for the VRPDR instances. It provides also the shortest average traveled distance over the 21 instances. For the average performance metric, our algorithm outperforms the other metaheuristics on 16 instances, while GA_{2-Opt} has the best value for 5 instances. None of the other algorithms has obtained a best value for this metric.

The improvement provided by our algorithm on average ranges between 3.51% and 9.75% compared to the tested metaheuristics. MAPSO_{2-Opt} is similar to the other metaheuristics in 6 instances. The average of the relative error for the best results is 1.56%.

5.2 Performance Assessment

In order to be able to compare our results accurately, we have also performed statistical analysis for a pairwise comparison of methods. For each comparison, the Wilcoxon signed rank test [9] shows that the differences among medians were statistically significant between MAPSO_{2-Opt} and the other algorithms (p-value < 0.05) at the 95% confidence level. In consequence, our algorithm performs better than all the other metaheuristics.

6 Conclusion and Future Work

We have presented a Multi-Adaptive Particle Swarm Optimization (MAPSO) approach for solving the Vehicle Routing Problem with Dynamic Requests (VRPDR). This approach has been tested on several instances of DVRP benchmarks. The experimental results show that MAPSO is able to find high quality solutions compared to other metaheuristics, and introduces new best solutions. The obtained results demonstrate the efficiency of our metaheuristic to solve this problem. For future work, we want to measure the adaptation process of our approach by using different features as accuracy of the obtained solutions and stability of the search process in a dynamic environment.

Acknowledgements. Authors acknowledge funds from the Associated Teams Program of the French National Institute for Research in Computer Science and Control (INRIA)(http://www.inria.fr).

References

1. Ai, T.J., Kachitvichyanukul, V.: A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. Comput. Oper. Res. 36(5), 1693–1702 (2009)
2. Alba, E.: Parallel metaheuristics: a New Class of Algorithms. Wiley Interscience, Hoboken (2005)
3. Bent, R., Van Hentenryck, P.: Online Stochastic and Robust Optimization. In: Maher, M.J. (ed.) ASIAN 2004. LNCS, vol. 3321, p. 286. Springer, Heidelberg (2004)
4. Blackwell, T., Branke, J.: Multi-swarm Optimization in Dynamic Environments. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 489–500. Springer, Heidelberg (2004)
5. Blackwell, T., Branke, J.: Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE transactions on evolutionary computation 10(4), 459–472 (2006)
6. Blackwell, T.: Particle swarm optimization in dynamic environments. In: Evolutionary Computation in Dynamic and Uncertain Environments, pp. 29–49. Springer, Berlin (2007)
7. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 3, pp. 1875–1882. IEEE Press, Washington (1999)

8. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Norwell (2001)
9. Conover, W.: Practical nonparametric statistics. Wiley, New York (1999)
10. Dantzig, G., Ramser, J.: The truck dispatching problem. Operations Research, Management Sciences 6(1), 80–91 (1959)
11. Hanshar, F., Ombuki-Berman, B.: Dynamic vehicle routing using genetic algorithms. Applied Intelligence 27, 89–99 (2007)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, vol. 4, pp. 1942–1948. IEEE Service Center, Piscataway (1995)
13. Kilby, P., Prosser, P., Shaw, P.: Dynamic VRPs: A study of scenarios. In: APES-06-1998, University of Strathclyde, U.K (1998)
14. Li, X., Branke, J., Blackwell, T.: Particle swarm with speciation and adaptation in a dynamic environment. In: GECCO 2006: Proceedings of the 8th annual conference on Genetic and Evolutionary Computation, pp. 51–58. ACM, New York (2006)
15. Lin, S.: Computer solutions of the traveling salesman problem. Bell System Computer Journal 44, 2245–2269 (1965)
16. Montemanni, R., Gambardella, L., Rizzoli, A., Donati, A.: A new algorithm for a dynamic vehicle routing problem based on ant colony system. Journal of Combinatorial Optimization 10, 327–343 (2005)
17. Oppacher, F., Wineberg, M.: The shifting balance genetic algorithm: Improving the GA in a dynamic environment. In: Proceedings of the Genetic and Evolutionary Computation Conference, vol. 1, pp. 504–510 (1999)
18. Parrott, D., Li, X.: A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. In: Congress on Evolutionary Computation (CEC 2004), vol. 1 (2004)
19. Psaraftis, H.: Dynamic vehicle routing: status and prospects. Annals of Opertations Reasearch 61, 143–164 (1995)
20. Rego, C.: Node-ejection chains for the vehicle routing problem: Sequential and parallel algorithms. Parallel Computing 27(3), 201–222 (2001)
21. Talbi, E.: Metaheuristics: from design to implementation. Wiley, Chichester (2009)
22. Ursem, R.: Multinational GAs: Multimodal optimization techniques in dynamic environments. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 19–26. Morgan Kaufmann, San Francisco (2000)
23. Yang, S.: Explicit memory schemes for evolutionary algorithms in dynamic environments. In: Evolutionary Computation in Dynamic and Uncertain Environments, pp. 3–28. Springer, Berlin (2007)

Off-line *vs.* On-line Tuning: A Study on $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for the TSP

Paola Pellegrini[1], Thomas Stützle[2], and Mauro Birattari[2]

[1] Dipartimento di Matematica Applicata
Università Ca' Foscari Venezia, Venezia, Italia
paolap@pellegrini.it
[2] IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{stuetzle,mbiro}@ulb.ac.be

Abstract. Stochastic local search algorithms require finding an appropriate setting of their parameters in order to reach high performance. The parameter tuning approaches that have been proposed in the literature for this task can be classified into two families: *on-line* and *off-line* tuning. In this paper, we compare the results we achieved with these two approaches. In particular, we report the results of an experimental study based on a prominent ant colony optimization algorithm, $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, for the traveling salesman problem. We observe the performance of on-line parameter tuning for different parameter adaptation schemes and for different numbers of parameters to be tuned. Our results indicate that, under the experimental conditions chosen here, off-line tuned parameter settings are preferable.

1 Introduction

The performance of stochastic local search (SLS) algorithms [15], depends on the appropriate setting of numerical and categorical parameters [7]. Methods that find good parameter settings in an automatic way have recently received strong attention by the research community [4,9,7,10,16,19]. A main contribution of those methods is to alleviate algorithm designers from the tedious and error-prone task of hands-on parameter adaptation.

The available approaches for automated parameter tuning can be classified into either *off-line* or *on-line* approaches. Off-line approaches exploit the knowledge gained in an *a priori* tuning phase, where parameter values are optimized based on a training set of instances. The algorithm is then deployed in a production phase with the selected parameter setting. Off-line approaches are typically black-box and they do not require any modification of the algorithm at hand. Examples of off-line approaches are F-Race [9], Iterated F-Race [3], CALIBRA [1] and ParamILS [16]. The main cost of off-line tuning is due to the use of resources in the *a priori* experimental phase.

This cost is avoided in on-line tuning approaches, which adapt the parameter values while solving an instance. An advantage of on-line tuning approaches is

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 239–250, 2010.

that they may adjust the parameter values to the characteristics of the particular instance that is being tackled. Hence, intuitively, on-line tuning approaches should benefit relative to off-line tuning when the instance class being tackled is more heterogeneous. In order to adjust the value of the parameters, on-line approaches often use either some search-based mechanism or a mechanism that is based on feedback from the search process. A particularly successful class of on-line algorithms are reactive search approaches as exemplified by reactive tabu search [4]. These approaches typically adapt very few key parameters of an algorithm and require substantial insight into algorithm behavior for their development. On-line parameter adaptation has also received a strong interest in the evolutionary computation community [19], where often general-purpose parameter adaptation schemes are studied.

In this paper, we compare the performance of off-line and on-line parameter tuning schemes on an ant colony optimization (ACO) algorithm. In particular, we study the application of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [24] to the traveling salesman problem (TSP). Our experimental setup is based on the initial conjectures that (i) for homogeneous instance sets off-line tuning should result in excellent performance; (ii) the higher is the number of parameters adapted the worse should get the performance of on-line tuned algorithms; and (iii) for heterogeneous instance sets on-line tuning should have an advantage over off-line.

As an off-line tuning method we use F-Race [5] on the set of candidate configurations we considered. The on-line tuning approaches are given the same set of candidate configurations and we test 5 on-line approaches. Each of the on-line approaches is tested for various numbers of parameters that are to be adapted online. Our results indicate that, in the setting considered, even if we knew a priori for all the possible subsets of parameters of equal cardinality the subset that results in the best performance, off-line tuning would remain the method of choice. In particular, in our example we can show that, when using off-line tuned parameters as initial values in on-line tuning, the performance of the latter worsens as the number of parameters to be adjusted increases.

2 $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System

$\mathcal{MM}$AS [24] is one of the most prominent ACO algorithms. It extends ant system [12] by a more aggressive pheromone update, the usage of upper and lower bounds on the range of possible pheromone trails, and few other details. For the experiments, we use the $\mathcal{MM}$AS implementation provided by the ACOTSP software [23]. In the experiments, we use as a local search the 2-opt algorithm. We refer the reader to the ACOTSP code and the original paper [24] for any detail on the characteristics of the algorithm. We shortly describe here the six parameters that we consider for tuning. The parameters include α and β, which weight the influence of the pheromone trail strength and the heuristic information, respectively, on the probability of choosing a specific edge in the solution construction; m, the number of ants in the colony; and ρ, which represents the pheromone evaporation rate. Here, ants use the pseudo-random proportional

action-choice rule of Ant Colony System [11], where with a probability q_0 an ant chooses deterministically, when being at a city i, the next city j as the one for which the product $\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}$ is maximal, where τ_{ij} and η_{ij} are the pheromone trail strength and the heuristic information, respectively. With a probability $1 - q_0$ the next city is chosen probabilistically as usual in $\mathcal{MM}$AS. A further parameter n indicates how many cities are considered as candidates for the next city.

3 Approaches for Off-line and On-line Tuning

For observing the impact of different tuning procedures on the performance of the algorithm, we consider one off-line and five on-line approaches.

For off-line tuning we apply F-Race [7,9]. F-Race takes as input a set of algorithm configurations and a sequence of instances. During the execution of F-Race, at each step, all configurations are run on one additional instance. After each step, configurations are discarded as they appear to be suboptimal on the basis of the available information. Thanks to this progressive elimination, F-Race uses all the available resources for focusing on the most promising configurations. For more details on F-Race we refer to [7].

The first on-line approaches tested follow the self-adaptive approach [13], that is, the determination of the parameter values is integrated into the algorithm's search process. This is done by associating pheromone trails to each possible value of a parameter and by using the ants' construction mechanism to choose which parameter value to adopt. After the solution construction, the pheromone update rule is applied to the trail associated to both parameter values and solution components. Various authors have proposed variants of such a self-adaptive approach [20,22,14,18]. The existing approaches differ mainly in two aspects: parameters can be associated either to each single ant or to the colony as a whole; and, parameters can be considered either independent from one another or as interdependent. In our study, parameters are treated as interdependent.

Typically, the self-adaptive approaches manage parameters at the ant-level, i.e., each ant selects its own parameter setting [20,22,14]. However, if each ant uses a different parameter setting, the speed-up techniques used in the ACOTSP code (essentially pre-computations of values required in the solution construction) cannot be used, leading to high computation time. Therefore we consider also the case in which parameters are managed at the colony-level, i.e., one parameter setting is fixed for all ants at the beginning of each iteration [18]. In this framework, we analyze three variants of the adaptive algorithm, that differ for the parameter values on which pheromone is deposited after the colony has completed its activity. In all cases, pheromone is deposited on the edges connecting the parameter values used in one specific iteration. In the first case, the parameter settings that receive reinforcement are either the ones of the current iteration or those used for generating the best-so-far solution. In the second case, the parameter settings reinforced are the ones for which the best solution was generated across all previous 25 iterations. In the third case, the parameter settings reinforced are the ones for which the best average solution cost was found

across all previous 25 iterations. When the adaptation is made at the colony-level, all six parameters described in Section 2 can be adapted ($q_0, \beta, \rho, m, \alpha, n$). When it is done at the ant-level, the on-line tuning can be applied only to the parameters involved in the solution construction (q_0, β, α, n).

The second on-line tuning mechanism we examine uses a search-based procedure for selecting the best values of parameters in the run of the algorithm [2]. The ant colony is split in groups of equal size; a parameter setting in the neighborhood of the incumbent one is assigned to each of them. The neighborhood of the current configuration is defined by all possible combinations that are obtained by increasing or decreasing the value of one parameter and keeping fixed all others. The configuration that corresponds to the best solution generated is used as the center of the neighborhood for the next iteration. Being parameters associated to groups of ants, the speed-up procedures used in the ACOTSP code cannot be fully exploited. With this mechanism, the four parameters involved in solution construction are adapted, that is, α, β, n, q_0.

All the five on-line tuning procedures have been implemented for the experimental analysis. For the sake of brevity we consider in the following only one of the self-adaptive approaches, the one that gives the best results. The whole analysis is available in [21].

4 Experimental Setup

We present an experimental analysis aiming at comparing the performance of off-line and on-line tuning under different experimental conditions. We consider four versions of $\mathcal{MM}$AS for the TSP, depending on the tuning procedure used:

- *literature* (L): parameter values are set as suggested in the literature [12]. These settings are highlighted in Table 1. This set of experiments is run as a baseline comparison;
- *off-line* (OFF): F-Race determines the values of the six parameters, which are then maintained fixed throughout all runs;
- *self-adaptive on-line* (SA): the self-adaptive mechanism at colony-level is used, starting from the parameter setting suggested in the literature [12];
- *off-line + self-adaptive on-line* (OFF+SA): the self-adaptive mechanism at colony-level is used, starting from the parameter setting returned by F-Race.

The same analysis has been done with the other adaptation schemes.

When an on-line approach is used, we solve each instance adapting alternatively one, two, ... , six parameters. In this way, we study how results change if the number of parameters adapted increases. Moreover, for each number of parameters adapted, we register the performance of the algorithm for all the possible combinations of parameters. In the rest of the paper, the name of all versions that include on-line tuning are followed by a number between parenthesis indicating how many parameters are adapted. The adaptation schemes are added on top of the ACOTSP software [23].

Table 1. Values that can be chosen for each parameter. The values reported in bold type are the ones suggested in the literature [12]. They are the values used in L setting.

parameter	values	parameter	values
α	0.5, **1**, 1.5, 2, 3	β	1, **2**, 3, 5, 10
ρ	0.1, **0.2**, 0.3, 0.5, 0.7 6	q_0	**0.0**, 0.25, 0.5, 0.75, 0.9
m	5, 10, **25**, 50, 100	n	10, **20**, 40, 60, 80

Table 2. Sets of instances considered. $U(a, b)$ indicates that for each instance of a set a number was randomly drawn between a and b. F-Race selection indicates the parameter settings selected by F-Race for a computation time limit of 10 CPU seconds.

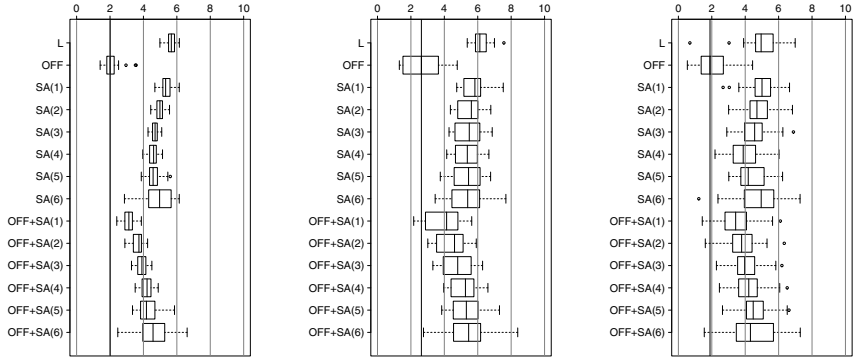
set	number of nodes	spatial distribution	F-Race selection					
			α	β	ρ	q_0	m	n
1	2000	uniform	1	5	0.75	0.5	25	20
2	2000	clustered	2	1	0.25	0.75	25	40
3	2000	uniform and clustered	1	1	0.25	0.9	25	20
4	$U(1000, 2000)$	uniform	1	5	0.75	0.25	50	20
5	$U(1000, 2000)$	clustered	2	2	0.25	0.75	50	40
6	$U(1000, 2000)$	uniform and clustered	1	1	0.25	0.9	50	20

For a fair comparison between off-line and on-line tuning, the same set of parameter values are available to the two approaches, that is, at each step the approaches can choose among a common set of parameter values. The possible values (used in this order in the self-adaptation scheme) are shown in Table 1.

We consider six sets of instances, all generated using portgen, the instance generator adopted in the 8th DIMACS Challenge on the TSP [17]. They differ in the number of cities included and in their spatial distribution, for details we refer to Table 2, where also the parameter values chosen by F-Race are indicated. We created these sets for having various levels of heterogeneity. The instance sets range from homogeneous sets where all instances are of a same size and a same spatial distribution of the nodes (either uniformly at random or clustered) to increasingly heterogeneous ones where the instances differ either in their size or also in the spatial distribution of the nodes; the most heterogeneous set is set 6.

For each set of instances, a separate run of F-Race is performed using 1000 training instances. The instances used for the tuning and the experimental phase are randomly selected, and the two sets are disjoint. All combinations of the values reported in Table 1 are considered as candidate settings. Hence, a total of 15,625 configurations is tested, on a maximum total number of runs equal to 156,250. The computation time available for each run is equal to the one considered in the experiments.

We executed experiments with two different termination criteria, 10 and 60 CPU seconds as measured on Xeon E5410 quad core 2.33GHz processors with 2x6 MB L2-Cache and 8 GB RAM, running under the Linux Rocks Cluster Distribution. The code is compiled with `gcc`, version 3.4. In 10 CPU seconds. In this environment, the ACOTSP code generates about 2 500-3 000 solutions for instances of set 1. The results presented in Section 5 depict the percentage error with respect to the optimal solution for 44 new test instances of each set. We performed one run on each instance for each parameter configuration [6,8].



(a) Set 1 (b) Set 3 (c) Set 6

Fig. 1. Results simulating no *a priori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one.

We analyzed the performance of each on-line version also considering as stopping criterion the construction of as many solutions as the *off-line* version. The results achieved are not qualitatively different from the ones obtained considering time as stopping criterion. In the following we show that on-line tuning is not convenient in the setting considered. This result is not due to the time overhead implied by adaptation, but due to the nature of the adaptation itself.

5 Experimental Results

The results for all sets of instances described in Section 4 and all the adaptation schemes described in Section 3 are reported in [21]. Due to the limited space available, we focus here on the results of the self-adaptation mechanism at colony-level for sets 1, 3, and 6: 2000 uniformly distributed nodes, 2000 nodes either clustered or uniformly distributed, and a random number of nodes between 1000 and 2000 either clustered or uniformly distributed. The on-line tuning approach shown here achieves the best results: the qualitative conclusions that can be drawn are very similar for all the adaptation schemes. We compare the two tuning approaches simulating different levels of knowledge on which are the most relevant parameters to be tuned on-line.

Experiment 1: no *a priori* knowledge on parameter importance for on-line tuning. If no *a priori* knowledge on parameter importance for on-line tuning is available, we use the average computed across all possible combinations for each number of parameters tuned as an indication of the expected performance level. These aggregate results are presented in Figure 1, where the boxplots summarize the average results in terms of percentage error.

The performance of OFF is the best for all sets of instances considered. The difference with respect to the *literature* version and to all *self-adaptive on-line*

ones is always statistically significant at the 95% confidence level, according to the Wilcoxon rank-sum test. The only exception is represented by OFF+SA(1) on set 2. Interestingly, the literature version (L) appears to be the worst for all sets. The reason is probably that the default literature settings have been developed for situations where the computation time available is rather large; this conjecture is confirmed in Experiment 4 on long run times.

Depending on whether L or OFF settings are used as initial parameter values, different behavior of the on-line parameter adaptation schemes can be observed. In the first case (results labeled SA in the plots), the on-line parameter adaptation schemes help to improve the reached solutions quality, the best being to adapt three or four parameters. Clearly, on-line tuning has some potential to improve upon fixed initial parameter values if these are not chosen appropriately. The result is very different in the second case, when starting from OFF parameter settings (results labeled OFF+SA in the plots). In this case, on-line tuning clearly worsens the final solution quality in a quite regular fashion. Interestingly, the more parameters are adapted, the worse is the final average solution quality reached. Remarkably, this conclusion remains the same for different levels of the heterogeneity of the instance sets.

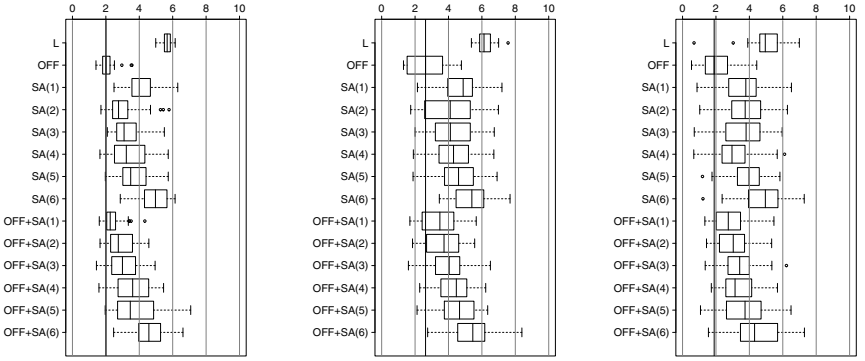
Experiment 2: perfect *a posteriori* knowledge on parameter importance for on-line tuning. Here we simulate the case in which the algorithm designer knows exactly which are the most important parameters to be tuned. This is done by considering the *a posteriori* best configuration for each cardinality of the subsets of parameters to be tuned. In other words, for each possible subset of one, two, ... , six parameters that are adapted on-line, we select the subset that results in the lowest average cost. Such a choice introduces a bias in favor of the on-line tuned versions, but, as shown below, the off-line version remains preferable. Hence, this does not invalidate the main conclusions of the analysis. The results of this best-case analysis are reported in Figure 2.

Interestingly, the off-line tuned version performs significantly better than most of the other versions but OFF+SA(2) for set 2. Hence, even for the most heterogeneous class of instances, set 6, OFF is performing better than the on-line tuned version. L is always significantly worse than all the other versions.

The quality of the final results, as a function of the number of parameters adapted, follows the same trend observed in Figure 1. It also confirms that a good starting point for the on-line tuning, as given by OFF, is preferable over a poor performing starting point, as given by L in this case.

Experiment 3: realistic *a priori* knowledge on parameter importance for on-line tuning. For understanding to which extent the best *a posteriori* configurations are those that one would actually test if she wished to adapt a given number of parameters, we asked six researchers and practitioners in the field of ACO to indicate their potential selection of the subset of parameters to be tuned on-line. The aggregated results are reported in Figure 3. We represent the average percentage error over the combinations of parameters suggested.

Obviously, OFF is the best performing version, given that it was already the best in the previous two experiments. For what SA is concerned, the results are



(a) Set 1 (b) Set 3 (c) Set 6

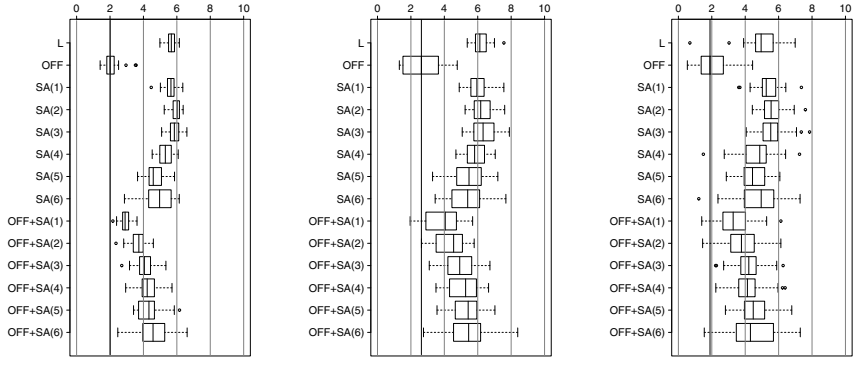
Fig. 2. Results simulating perfect *a posteriori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one.

close to the ones observed when considering the average over all combinations. In particular, the variance of the distribution of the percentage error is quite low, and the quality of the solution is comparable to the one achieved by the *literature* version. When we consider OFF+SA, the results of the survey lead to solutions that are in between the average and the *a posteriori* best configuration. Let us remark that the difference between these two representations of the results is in this case quite moderate. Thus, if a well performing initial parameter setting is used, the intuition on the set of parameters that is convenient to be adapted can be expected to lead close to the best possible results.

Experiment 4: long runs. In this experiment, we examine the impact of the termination condition on the results. In particular, we executed the same set of experiments on the instances of set 1 for a maximum CPU time of 60 seconds (instead of the previously used 10 seconds). The rationale of these experiments is to give the on-line tuning mechanism a longer time to adjust parameters. Figure 4 reports the results achieved using the three cases of no *a priori* information, perfect *a posteriori* information, and realistic *a priori* information, which have been examined in the previous three experiments.

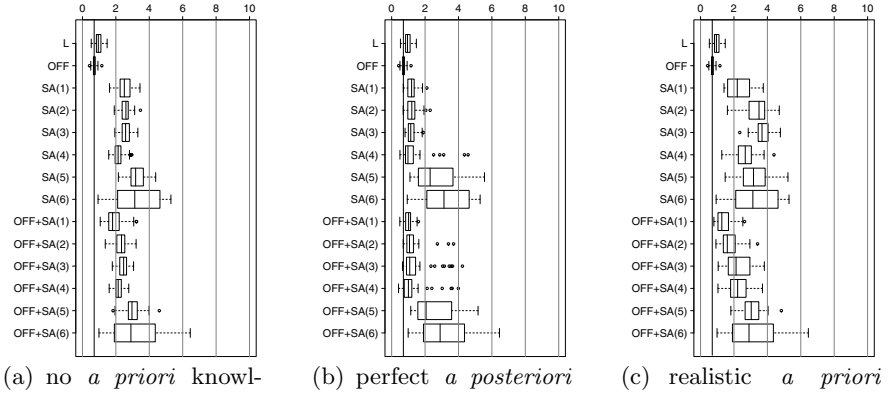
The conclusions that can be drawn are very much in line with those for the shorter computational time: Off-line tuning allows $\mathcal{MM}$AS to achieve the best performance with respect to all the other versions. The differences are statistically significant (checked using the Wilcoxon rank-sum test). A major improvement is experienced by the *literature* version: as we expected, longer runs allow the parameter setting suggested in [12] to achieve good results.

The relative performance of the on-line tuned versions with respect to OFF and L slightly improves. When considering the effect of on-line tuning averaged across all subsets of parameters that are adapted, the effect of different cardinalities of these subsets reflects quite closely the above observations: we cannot



(a) Set 1 (b) Set 3 (c) Set 6

Fig. 3. Results simulating realistic *a priori* knowledge on parameter importance for on-line tuning. Runs of 10 seconds. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one.



(a) no *a priori* knowledge (b) perfect *a posteriori* knowledge (c) realistic *a priori* knowledge

Fig. 4. Results in long runs. Runs of 60 seconds. Instances of set 1. The horizontal axis represents the percentage error. The different versions tested are listed on the vertical one.

identify a clear trend for SA, while for OFF+SA we note that the results get worse as the number of parameters that are adapted on-line increases. If we suppose perfect *a posteriori* information, the behavior of SA and OFF+SA is much improved (Figure 4(b)). The results are always significantly worse than OFF, while in some cases they become comparable to L: we have not observed any statistically significant difference between L and SA(1), SA(4) and OFF+SA(4); the difference is significant in favor L in all other cases.

Observing the results of the survey, representing realistic *a priori* knowledge, (Figure 4(c)), we can notice that the performance of the selected configurations

in some cases are even worse than the overall average (Figure 4(a)). This happens for SA(2), SA(3), SA(4), OFF+SA(4), and OFF+SA(5). This observation strengthens the claim that tuning on-line just one or two parameters, instead of a large number, is the most convenient choice: Not only we can expect the approach to achieve quite good results (in some cases not worse than off-line tuning), but also we can assume that our intuition allows us to choose the parameters to adapt so that the potential of the tuning is actually exploited.

Summary of results. From the results just described we can conclude that:

- off-line tuning performs better than on-line tuning under all the experimental conditions tested;
- it is preferable to apply on-line tuning to few parameters than to many;
- if the initial parameter setting is a well-behaving configuration, our intuition on the most important parameters to adapt allows to exploit the on-line tuning more efficiently than if the setting suggested in the literature is used as starting configuration.

The heterogeneity of the class of TSP instances tackled does not appear to have a strong impact on the relative performance of the different versions.

6 Conclusions

In this paper we have compared the results achieved by $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System when its parameters are tuned off-line and when they are tuned on-line.

We have proposed an experimental analysis based on one off-line tuning and five on-line tuning procedures. We have considered $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for the TSP, and we have solved instances of six sets, differing in the heterogeneity of the instances included. Within this setting we have tested three main conjectures on the quality of the results achievable by tuning parameters off-line *vs.* on-line: two of them have been confirmed by the experimental evidence, while one has actually been contradicted. In particular, (i) as expected, for homogeneous instance sets off-line tuning has resulted in excellent performance; (ii) as expected, the higher the number of parameters adapted the worse the performance of on-line tuned algorithms; and (iii) contrarily to what expected, for heterogeneous instance sets on-line tuning has not had an advantage over off-line: also on these instances off-line tuning has resulted in excellent performance. In this paper we have reported the results achieved by the best one, while the complete analysis is shown in [21]. The conclusions that can be drawn are equivalent regardless the specific approach considered.

These conclusions need to be tested on other combinatorial optimization problems. The merits of on-line tuning, for example, may emerge if the instances to be tackled are extremely different from each other, as it is the case for some scheduling problems. Further research will be performed in this direction.

In the cases in which on-line tuning may be advantageous, the results reported suggest that the implementation of a hybrid between the off-line and on-line

approach may be very promising: parameters may be first tuned off-line, and then one or two of them may be adapted while solving each instance. In this way, high quality solutions may be found. Thanks to a social experiment, we could observe that researchers' intuition on the most important parameters to tune on-line allows to get better results if an optimized parameter setting is used for starting the adaptation, rather than if the default setting is used.

A further possible direction of future research consists in using an off-line tuning approach for setting the value of the meta-parameters that drive the adaptation in on-line tuning. Examples of such meta-parameters are the number of iterations used in the self-adaptation scheme with multiple-colony comparison, or the number of neighbor-configurations in the search-based adaptation.

Acknowledgments. This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium. Mauro Birattari and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are Research Associates. The authors thank the colleagues that answered the survey described in the paper.

References

1. Adenso-Díaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental designs and local search. Operations Research 54(1), 99–114 (2006)
2. Anghinolfi, D., Boccalatte, A., Paolucci, M., Vecchiola, C.: Performance evaluation of an adaptive ant colony optimization applied to single machine scheduling. In: Li, X., Kirley, M., Zhang, M., Green, D., Ciesielski, V., Abbass, H.A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K.C., Branke, J., Shi, Y. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 411–420. Springer, Heidelberg (2008)
3. Balaprakash, P., Birattari, M., Stützle, T.: Improvement strategies for the F-race algorithm: Sampling design and iterative refinement. In: Bartz-Beielstein, T., et al. (eds.) HM 2007. LNCS, vol. 4771, pp. 108–122. Springer, Heidelberg (2007)
4. Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intelligent Optimization. Operations Research/Computer Science Interfaces, vol. 45. Springer, Berlin (2008)
5. Birattari, M.: Race. R package (2003), http://cran.r-project.org
6. Birattari, M.: On the estimation of the expected performance of a metaheuristic on a class of instances. How many instances, how many runs? Tech. Rep. TR/IRIDIA/2004-01, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2004)
7. Birattari, M.: Tuning Metaheuristics: A Machine Learning Perspective. Springer, Berlin (2009)
8. Birattari, M., Dorigo, M.: How to assess and report the performance of a stochastic algorithm on a benchmark problem: Mean or best result on a number of runs? Optimization Letters 1(3), 309–311 (2007)
9. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W., et al. (eds.) GECCO 2002, pp. 11–18. Morgan Kaufmann Publishers, San Francisco (2002)
10. Coy, S., Golden, B., Runger, G., Wasil, E.: Using experimental design to find effective parameter settings for heuristics. Journal of Heuristics 7(1), 77–97 (2001)

11. Dorigo, M., Gambardella, L.M.: Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
12. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
13. Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E.: Parameter control in evolutionary algorithms. In: [19], pp. 19–46
14. Förster, M., Bickel, B., Hardung, B., Kókai, G.: Self-adaptive ant colony optimisation applied to function allocation in vehicle networks. In: GECCO 2007, pp. 1991–1998. ACM Press, New York (2007)
15. Hoos, H.H., Stützle, T.: Stochastic Local Search—Foundations and Applications. Morgan Kaufmann Publishers, San Francisco (2005)
16. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research 36, 267–306 (2009)
17. Johnson, D., McGeoch, L., Rego, C., Glover, F.: 8th DIMACS implementation challenge (2001), http://www.research.att.com/~dsj/chtsp/
18. Khichane, M., Albert, P., Solnon, C.: A reactive framework for ant colony optimization. In: Stützle, T. (ed.) LION 3. LNCS, vol. 5851, pp. 119–133. Springer, Heidelberg (2009)
19. Lobo, F., Lima, C.F., Michalewicz, Z.: Parameter Setting in Evolutionary Algorithms. Springer, Berlin (2007)
20. Martens, D., Backer, M.D., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B.: Classification with ant colony optimization. IEEE Transactions on Evolutionary Computation 11(5), 651–665 (2007)
21. Pellegrini, P., Stützle, T., Birattari, M.: Companion of off-line and on-line tuning: a study on $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for TSP (2010) IRIDIA Supplementary page, http://iridia.ulb.ac.be/supp/IridiaSupp2010-008/
22. Randall, M.: Near Parameter Free Ant Colony Optimisation. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 374–381. Springer, Heidelberg (2004)
23. Stützle, T.: ACOTSP: A software package of various ant colony optimization algorithms applied to the symmetric traveling salesman problem (2002), http://www.aco-metaheuristic.org/aco-code
24. Stützle, T., Hoos, H.H.: MAX–MIN ant system. Future Generation Computer Systems 16(8), 889–914 (2000)

Opinion Dynamics for Decentralized Decision-Making in a Robot Swarm

Marco A. Montes de Oca, Eliseo Ferrante, Nithin Mathews,
Mauro Birattari, and Marco Dorigo

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{mmontes,eferrant,nmathews,mbiro,mdorigo}@ulb.ac.be

Abstract. In this paper, we study how an opinion dynamics model can be the core of a collective decision-making mechanism for swarm robotics. Our main result is that when opinions represent action choices, the opinion associated with the action that is the fastest to execute spreads in the population. Moreover, the spread of the best choice happens even when only a minority is initially advocating for it. The key elements involved in this process are consensus building and positive feedback. A foraging task that involves collective transport is used to illustrate the potential of the proposed approach.

1 Introduction

Large groups of animals can exhibit behavioral patterns that resemble the ones observed in physical systems that are composed of many simple entities [4,3]. This observation has led to the development of *opinion dynamics models*, which are used to study large-scale social, economic, and natural phenomena that involve many interacting agents [3].

In this paper, we study how an opinion dynamics model can be the core of a collective decision-making mechanism for swarm robotics. The opinion dynamics model used in our work, originally proposed by Krapivsky and Redner [8], allows a large population of agents to reach consensus on one of two alternatives. We are interested in situations where opinions represent actions that take time to perform. Our goal is to determine if, and under which conditions, the action selected by the swarm is the one that takes less time on average to perform. Such outcome would be useful in swarm robotics applications in which the number of times a particular action is selected is correlated with the amount of work performed, thus maximizing the productivity of a robot swarm.

In Krapivsky and Redner's model the *majority rule* is repeatedly applied on teams of three agents. With the majority rule, team members adopt the opinion shared by the majority of the team members. In addition to the majority rule, we also use the *expert rule*, through which team members adopt the opinion of a single agent, called the expert, which is selected according to some problem-specific criterion. As these rules are at the opposite ends of the spectrum of all weighted majority rules [1], our study allows us to have an intuition of the results that would be obtained if a weighted majority rule was used.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 251–262, 2010.

The main result of our study is that the dynamics of the system makes the swarm select with high probability the action that is the fastest to execute. When using the expert rule, the fastest-to-execute action is selected by the swarm even when only a minority of the robots is initially in favor of it. The potential of the proposed approach as a decentralized decision-making mechanism for swarms of robots is illustrated through a foraging task that involves collective transport. As a result of the application of the proposed approach, the swarm of robots is able to select the shortest path without requiring the robots to measure path-travel times or to rely on pheromone-like information. Other applications where it is desired to select automatically the choice that increases the efficiency of a system composed of many interacting agents could benefit from the use of the mechanism presented in this paper.

The rest of the paper is structured as follows. In Section 2, we describe related work in the area of decentralized decision-making in swarms of robots. In Section 3, we describe the model and the decision rules used. In Section 4, we describe the task and the experimental setup used to evaluate the effectiveness of the proposed approach. Results are presented in Section 5. Conclusions and future work are given in Section 6.

2 Related Work

Many decentralized decision-making mechanisms have been inspired by the behavior of insects. For example, the pheromone-laying and pheromone-following behavior of some ant species [7] has inspired many works. Most of them have focused on the simulation of pheromones through the use of chemical substances [16], by projecting images on the ground [18], by deploying RFID tags [9], or by using robots as message-relay devices or as beacons so as to form robot chains [19,14,11,12]. Recently, pheromones have been simulated by signals sent by robots that belong to a swarm different from the one engaged in the foraging task [5]. These mechanisms have some important disadvantages. For example, designing sensors for detecting chemicals reliably is a very difficult task. Using robots as beacons requires the development of complex controllers to allow an individual robot to play different roles both within and outside a robot chain. Using RFID tags requires the modification of the environment prior to the deployment of the swarm, which is impossible in some cases. Projected pheromones can be impractical because a central computer is needed. In contrast to all these works, our system does not require the explicit simulation of pheromones.

Other behaviors of insects have also been a source of inspiration. For example, trophallaxis, which is the insect-to-insect exchange of food, has been the inspiration for a distributed mechanism to create a gradient in the environment to help robots navigate [17]. The aggregation behavior exhibited by cockroaches has been the source of inspiration for a site-selection mechanism with robots [6]. The best-of-N selection mechanism proposed by Parker and Zhang [13] has been inspired by the nest-selection mechanism used by some species of ants. To the best of our knowledge, this last work is the most similar to ours. In both of them,

a collective decision is the result of the competition between alternatives. The main difference, however, lies in the consensus building mechanism. In Parker and Zhang's approach, robots need to know whether there is a sufficient number of robots in favor of one alternative before committing to it. Robots do that through a quorum test that depends on a parameter that the designer needs to set before deployment. This is a critical issue because the first alternative that is identified as dominant through the quorum test will be the alternative chosen by the swarm. In our work, the collective decision is the result of self-organization.

Another work related to ours is the one of Wessnitzer and Melhuish [20]. In their work, robots have to chase and immobilize two "prey." Robots capture one prey after the other. To select which prey to immobilize, robots apply the local majority rule to break the symmetry of the decision problem and to make the population agree on one choice only. Our work goes a step further by considering the effects of implicit time costs in the robots' actions.

3 Opinion Dynamics and Decentralized Decision-Making

In this section, we describe the model and the decision rules used in our study. We also explain how this model becomes the core of a decentralized decision-making mechanism when opinions represent actions that take time to perform.

3.1 Opinion Dynamics Model and Decision Rules

We use the opinion dynamics model proposed by Krapivsky and Redner [8]. It operates on a population of N agents, each of which can be in one of two possible states, called opinions. The system evolves as follows: A team of three agents is picked at random without replacement from the population. Then, the individual opinions of the team members are aggregated and transformed into a team opinion by a decision rule. After this, all team members adjust their individual opinions to match the resulting team opinion. The team members are put back in the population and a new team is picked. The process is repeated until all agents share the same opinion.

We use two decision rules: the majority rule (the one studied in [8]), and the expert rule. When the majority rule is used, all team members will assume the opinion that at least two team members share. With the expert rule, agents will assume the opinion of a single agent, called the expert (See Section 4.2 for information on the criterion used to choose which agent plays this role). Figure 1 shows an example of the application of the majority and of the expert rules.

3.2 Opinion Dynamics, Actions, and Robots

The opinion dynamics model described above can be used as the basis of a decentralized decision-making mechanism in a swarm of robots. Three elements need to be taken into account to do it: (i) opinions need to be interpreted as actions that robots execute, (ii) actions take time to perform, and (iii) robots can

(a) Majority rule (b) Expert rule

Fig. 1. Example application of the majority and the expert rules on a team of three agents with different opinions (represented by different color shades). Figure (a) shows the outcome of the majority rule: the team adopts the opinion of the majority. Figure (b) shows the outcome of the expert rule: the team adopts the opinion of the expert (marked with a × symbol).

operate in parallel. These elements are modeled through a parallel version of the opinion dynamics model described in Section 3.1. Instead of picking at random one team of three agents, we pick k teams without replacement. Then, each team selects an opinion according to a decision rule and its members adopt that opinion. The actions associated with the adopted opinions are then executed. The execution time associated with an action is not necessarily the same from one execution to another due to unexpected events during execution (e.g., an obstacle may have to be avoided, or a robot skids while trying to move). When a team finishes executing an action, its members become available again to form another team. The new team cannot have as members robots that are at that moment executing an action. The process continues until the swarm reaches consensus, the time allocated for the task is over, or the demand for the task ceases to exist. Figure 2 shows an example of the process just described.

4 Evaluation Scenario and Setup

In this section, we describe the task, the simulation environment, and the experimental setup used to evaluate the effectiveness of the proposed approach.

4.1 The Task

We chose a foraging task that involves collective transport as an example of the kind of applications the proposed approach could be used for. The environment consists of a storage room and two resource rooms in which there are objects of interest. There are two kinds of robots: robots that can manipulate the objects of interest but that cannot move by themselves, and robots that cannot manipulate the objects of interest but that can move autonomously and carry the manipulator robots. The task is to collect as many objects of interest as possible within some time limit from the two resource rooms, and deposit them in the storage room. To accomplish the task, the robots that can move autonomously

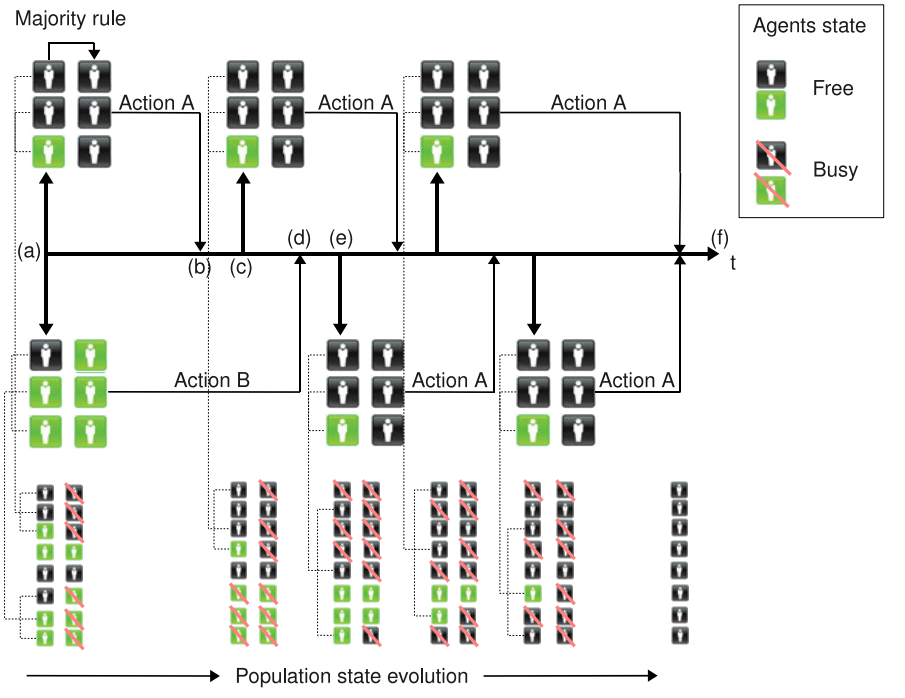
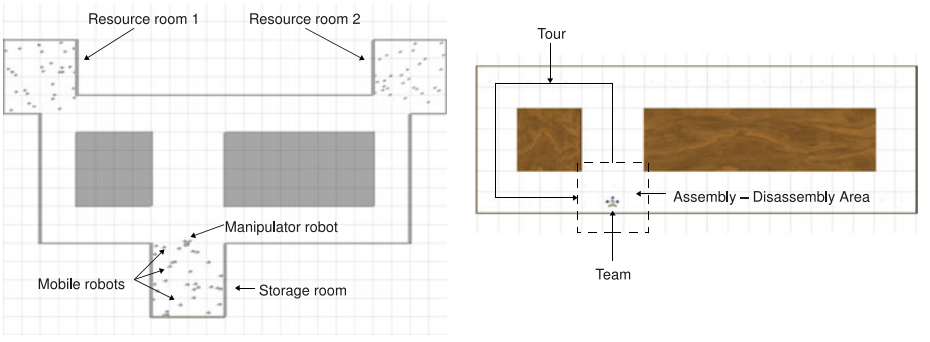


Fig. 2. Example of the dynamics induced by the majority rule on a population of 8 agents with 2 teams of 3 agents each. The opinion represented in black is associated with action A, while the opinion represented in light color is associated with action B. Action A is faster to execute than action B, on average. This system evolves as follows. First, in point (a), two teams are formed at random and the majority rule is applied on each one of them. Each team then executes the selected action. In point (b), a team finishes. In point (c) a new team is formed from the set of free agents (busy agents are not considered when the selection occurs). The time it takes to form a team is represented by the distance between points (b) and (c). After the application of the majority rule in (c), the team performs the agreed action (action A, in this case). In point (d), the other team finishes and a new team is formed (point (e)). Again, the majority rule is applied once more to decide which action to perform (action A, again). The process continues until the population reaches consensus (point (f)). In this example, the population changes from a heterogeneous opinion state to a homogeneous one that corresponds to the fastest-to-execute action.

must form small teams to carry the manipulator robots. These teams of robots go back and forth between the storage and resource rooms. It is in general desirable to go to the closest resource room in order to maximize the number of collected objects in a given amount of time. The path that leads to the closest resource room is a priori unknown to the robots. Figure 3(a) shows a picture of the complete envisioned scenario.

In the complete scenario, the choice that robots face is whether to turn left or right to go to a resource room. To focus only on the decision-making aspect of



(a) Complete scenario (b) Simplified scenario

Fig. 3. Test task. Figure (a) shows a complete view of the scenario. A swarm of robots must collect as many objects as possible within some time limit from the two resource rooms. Figure (b) shows a simplified version of the scenario. Robots must choose between making the left or right tour. The simplified scenario allows us to concentrate on the decision-making aspect of the task. See text for more information.

the task and not on other technicalities, such as object manipulation, collective obstacle avoidance, or self-assembly, we use a simplified version of the scenario described above, which is shown in Figure 3(b). Instead of a storage room there is an assembly-disassembly area where the mobile robots attach to and detach from the manipulator robots. Navigation is possible thanks to visual aids similar to the ones used in [12]. In this simplified scenario, the choice faced by the robots is simply to either turn left or right. These choices represent the robots' opinions on which the decision-making mechanism will operate. The execution of a chosen action starts with the formation of a team of robots, it continues with the navigation around the big obstacle in the chosen direction, and finishes once a complete lap is made.

4.2 Setup

All the experiments reported in this paper were performed in simulation. The simulator that we used [15] was developed for the *SWARMANOID* project.[1] This simulator uses the Open Dynamics Engine library[2] to simulate accurately physical interactions with the environment and between robots. The robot models are based on the physical and electronic design of the actual *SWARMANOID* robots (currently under development). The mobile robot model is based on the *Foot-bot*, and the manipulator robot model is based on the *Hand-bot* [2].

We ran simulations with different swarm sizes ($N \in \{8, 16, 32, 64\}$), different initial opinion biases (according to a parameter $p \in \{0.05, 0.1, \ldots, 0.95\}$, which is the probability of a robot's initial opinion to advocate for the left path), different

[1] http://www.swarmanoid.org/
[2] http://www.ode.org/

numbers of teams ($k \in \{2, 4, 8, 16\}$), and different action-execution time ratios ($r \in \{1, 2, 4\}$). The action-execution time ratio is defined as $r = l_{right}/l_{left}$, where l_{left}, and l_{right} are the length of the left and right paths, respectively. The reference length, l_{left}, was adjusted so that no collisions between teams of robots occurred. This was done by making the length of the left path much greater than the total length that results from lining up the k teams used in a given experiment. Teams had 3 robots each. A simulation was run until the swarm achieved consensus. 100 trials were run for each combination of parameters.

An extra parameter of the system is the criterion used to select the expert when applying the local expert rule. In our experiments, that criterion is the absolute number of completed laps (no distinction between paths is made), that is, the most "experienced" robot is selected as the expert. The study of other criteria to select the expert is left for future work.

5 Results

We are interested in two aspects of the system: (i) the probability with which the swarm selects the fastest-to-execute action as a function of the initial opinion bias, and (ii) the number of team formations needed to reach consensus on one of the alternative choices as a function of the initial opinion biases and of the swarm size. For both aspects, we evaluate the effects of different numbers of teams, k, and of different action-execution time ratios, r.

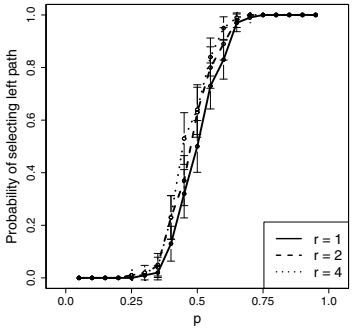
5.1 Probability of Selecting the Fastest-to-Execute Action

Figure 4 shows the estimated probability of selecting the left path, that is, the fastest-to-execute action, as a function of the initial opinion bias in a swarm of $N = 64$ robots.[3] We estimate this probability by dividing the number trials the system reached consensus on the left path by the total number of trials.

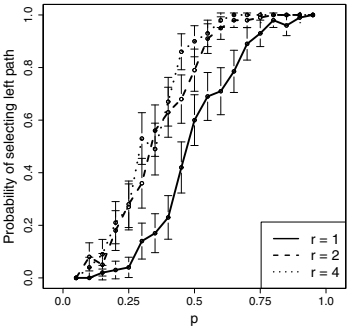
There is a nonlinear relationship between the initial opinion bias and the final probability with which the swarm chooses an alternative. In all cases, there is a critical bias p_c such that if $p < p_c$ the swarm will choose one opinion, and if $p > p_c$ the swarm will choose the other opinion. In the case where $r = 1$, that is, when there is no difference between the alternative choices, the critical initial bias is $p = 0.5$. When $r > 1$, $p_c < 0.5$ for both decision rules. With the majority rule, the higher the action-execution time ratio, the lower the critical bias. Furthermore, the critical bias decreases as the number of teams active in the environment increases (the critical bias is lower when $k = 16$ than when $k = 4$). With the expert rule, the critical bias is, in general, lower than with the majority rule; however, the actual value depends more strongly on the number of teams than on the action-execution time ratio.

From a practical point of view, a small critical bias is desirable because it means that the action that is fastest to execute will be selected by the whole swarm even when only a minority of the agents is initially in favor of it. In this

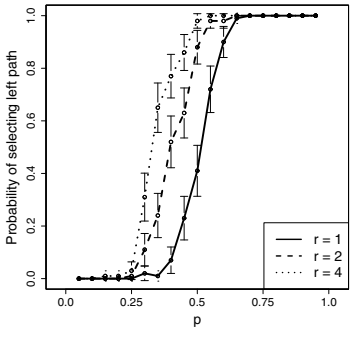
[3] We refer the reader to [10] for the complete set of results.



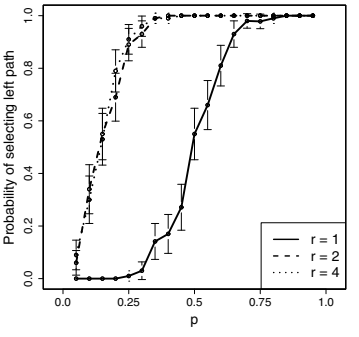
(a) Majority rule: $N = 64$, $k = 4$



(b) Expert rule: $N = 64$, $k = 4$



(c) Majority rule: $N = 64$, $k = 16$



(d) Expert rule: $N = 64$, $k = 16$

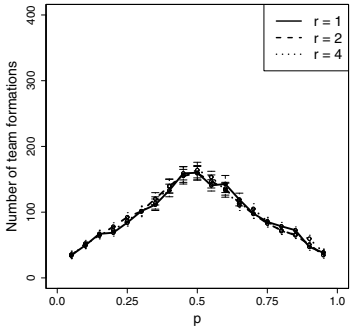
Fig. 4. Probability of selecting the fastest-to-execute action as a function of the initial opinion bias in a swarm of 64 robots with 4 and 16 teams. Figures (a) and (c) show the results obtained with the majority rule. Figures (b) and (d) show the results obtained with the expert rule. Error bars indicate the confidence interval at a 95% level.

sense, the expert rule is the best suited for this purpose because it can spread more easily the opinion of the minority. For example, see Figure 4(d), when $r > 1$. In this case, the swarm of 64 robots chooses the left path with probability 0.8 when just 13 robots (20% of the swarm) initially choose it.

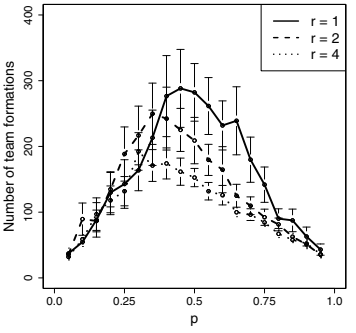
Small swarms ($N < 32$ with $k < 8$) have greater difficulties than large swarms in detecting differences in the average action-execution times (results shown in [10]). This may be due to the rapid opinion fluctuations that are amplified by the system. For instance, in an 8-robot swarm, one robot represents the 12.5% support for one or another opinion. Consensus is reached, but the chosen alternative is random.

5.2 Number of Team Formations Needed to Reach Consensus

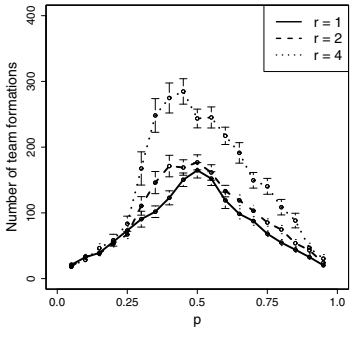
The number of team formations needed to reach consensus (NTFC) depends on the initial opinion bias (see Figure 5). In all cases, the maximum NTFC is reached when the initial opinion bias is equal to the critical bias (see Section 5.1).



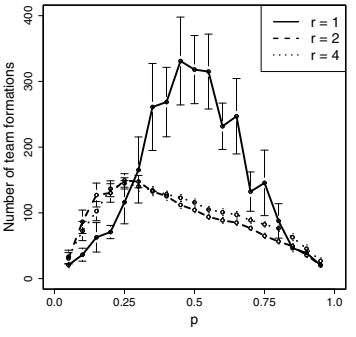
(a) Majority rule: $N = 64$, $k = 4$



(b) Expert rule: $N = 64$, $k = 4$



(c) Majority rule: $N = 64$, $k = 16$

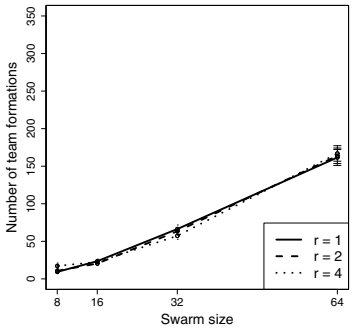


(d) Expert rule: $N = 64$, $k = 16$

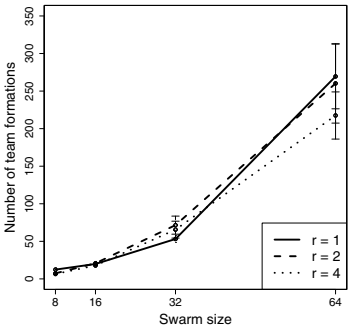
Fig. 5. Number of team formations needed to reach consensus as a function of the initial opinion bias. These results correspond to the case where $N = 64$. Figures (a) and (c) show the results obtained with the majority rule. Figures (b) and (d) show the results obtained with the expert rule. Error bars indicate the confidence interval at a 95% level.

The NTFC using either the majority rule, or the expert rule, depends on the number of teams that are active in the environment (k) and the action-execution time ratio (r). With the majority rule, the NTFC increases as k and r increase. With the expert rule, the NTFC decreases as k and r increase. In conclusion, with the expert rule the opinion associated with the fastest action is spread more rapidly than with the majority rule, especially in the presence of large action-execution time ratios.

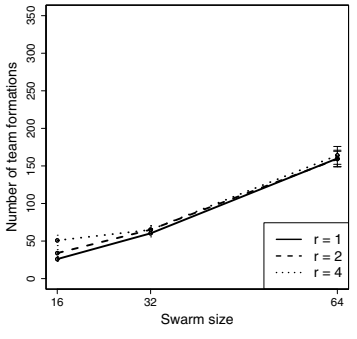
In case there is no a priori information about the quality of the alternatives the robots must choose from, the most reasonable strategy to initialize the system is to have a balanced initial opinion bias, that is, $p = 0.5$. Under that circumstance, one may ask what would be the NTFC as a function of the swarm size. The answer is shown in Figure 6. Not surprisingly, the NTFC increases with the swarm size in all cases. With the majority rule, there appears to be no significant difference in the NTFC if the number of active teams in the environment changes.



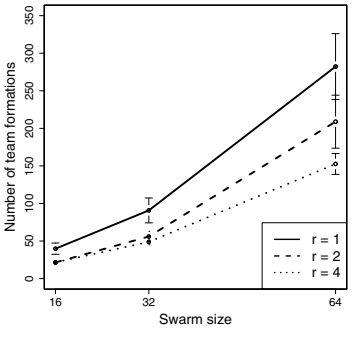
(a) Majority rule: $p = 0.5$, $k = 2$



(b) Expert rule: $p = 0.5$, $k = 2$



(c) Majority rule: $p = 0.5$, $k = 4$



(d) Expert rule: $p = 0.5$, $k = 4$

Fig. 6. Number of team formations needed to reach consensus as a function of the swarm size. These results correspond to the case where $p = 0.5$. Figures (a) and (c) show the results obtained with the majority rule. Figure (b) and (d) show the results obtained with the expert rule. Error bars indicate the confidence interval at a 95% level.

With the expert rule, however, if more teams are deployed and the swarm size increases, the NTFC decreases.

6 Conclusions and Future Work

We presented a decentralized decision-making mechanism for swarm robotics systems whose main elements are: (i) a consensus-building mechanism based on an opinion dynamics model, and (ii) actions that take different average time to execute. The consensus-building mechanism can be used with any weighted majority rule. In this paper, we explored the two extremes: the simple majority rule and the expert rule. We showed how the dynamics of the system allows the swarm to choose with a high probability the action that is the fastest to execute. With the expert rule, the fastest-to-execute action is selected by the whole swarm even when only a minority of the agents is initially in favor of it.

The difference in the execution time of the alternative actions induces a positive feedback process that is ultimately the responsible for the swarm's final choice. This phenomenon has parallels with the decision-making mechanism exploited by ants [7]. However, in our system agents influence each other by forming teams instead of doing it via pheromones.

From a practical point of view, forming small teams allows the number of messages between robots to be kept to a minimum. As a consequence, the system is able to scale up to many robots (in fact, the more robots, the better the system performs) without having to deal with interference or bandwidth problems.

Besides porting the system to real robots, future work includes extending the mechanism to sequential, adaptive and multi-choice decision-making. A first step in this direction will be to switch to a multidimensional probabilistic representation of a robot's opinion.

Acknowledgments. This work was partially supported by the SWARMANOID project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission, under grant IST-022888, and by the VIRTUAL SWARMANOID project funded by the Fund for Scientific Research F.R.S.-FNRS of Belgium's French Community. The information provided is the sole responsibility of the authors and does not reflect the European Commission's opinion. The European Commission is not responsible for any use that might be made of data appearing in this publication. M. Dorigo and M. Birattari acknowledge support from the F.R.S.-FNRS of Belgium's French Community, of which they are a research director and a research associate, respectively. We thank A. Brutschy and C. Pinciroli for their useful advice.

References

1. Berend, D., Chernyavsky, Y.: Ranking of decision rules with random power distribution. Mathematical and Computer Modelling 48(9-10), 1326–1334 (2008)
2. Bonani, M., Magnenat, S., Rétornaz, P., Mondada, F.: The hand-bot, a robot design for simultaneous climbing and manipulation. In: Xie, M., Xiong, Y., Xiong, C., Liu, H., Hu, Z. (eds.) ICIRA 2009. LNCS, vol. 5928, pp. 11–22. Springer, Heidelberg (2009)
3. Castellano, C., Fortunato, S., Loreto, V.: Statistical physics of social dynamics. Reviews of Modern Physics 81(2), 591–646 (2009)
4. Chakrabarti, B., Chakraborti, A., Chatterjee, A. (eds.): Econophysics and sociophysics: Trends and perspectives. Wiley-VCH, Wienheim (2006)
5. Ducatelle, F., Di Caro, G., Gambardella, L.M.: Cooperative self-organization in a heterogeneous swarm robotic system. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2010), pp. 87–94. ACM Press, New York (2010)
6. Garnier, S., Gautrais, J., Asadpour, M., Jost, C., Theraulaz, G.: Self-organized aggregation triggers collective decision making in a group of cockroach-like robots. Adaptive Behavior 17(2), 109–133 (2009)
7. Goss, S., Aron, S., Deneubourg, J.L., Pasteels, J.M.: Self-organized shortcuts in the argentine ant. Naturwissenschaften 76(12), 579–581 (1989)

8. Krapivsky, P.L., Redner, S.: Dynamics of majority rule in two-state interacting spin systems. Physical Review Letters 90(23), 238701.1–238701.4 (2003)
9. Mamei, M., Zambonelli, F.: Physical deployment of digital pheromones through RFID technology. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), pp. 1353–1354. ACM Press, New York (2005)
10. Montes de Oca, M.A., Ferrante, E., Mathews, N., Birattari, M., Dorigo, M.: Opinion dynamics for decentralized decision-making in a robot swarm: Complete data (2010), Supplementary information page at http://iridia.ulb.ac.be/supp/IridiaSupp2010-004/
11. Nouyan, S., Campo, A., Dorigo, M.: Path formation in a robot swarm: Self-organized strategies to find your way home. Swarm Intelligence 2(1), 1–23 (2008)
12. Nouyan, S., Groß, R., Bonani, M., Mondada, F., Dorigo, M.: Teamwork in self-organized robot colonies. IEEE Transactions on Evolutionary Computation 13(4), 695–711 (2009)
13. Parker, C.A.C., Zhang, H.: Cooperative decision-making in decentralized multiple-robot systems: The best-of-N problem. IEEE/ASME Transactions on Mechatronics 14(2), 240–251 (2009)
14. Payton, D., Daily, M., Estowski, R., Howard, M., Lee, C.: Pheromone robotics. Autonomous Robots 11(3), 319–324 (2001)
15. Pinciroli, C.: Object Retrieval by a Swarm of Ground Based Robots Driven by Aerial Robots. Mémoire de DEA, Université Libre de Bruxelles, Bruxelles, Belgium (2007)
16. Russell, R.A.: Ant trails – An example for robots to follow? In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 1999), pp. 2968–2703. IEEE Press, Piscataway (1999)
17. Schmickl, T., Crailsheim, K.: Trophallaxis within a robotic swarm: Bio-inspired communication among robots in a swarm. Autonomous Robots 17(2), 109–133 (2009)
18. Sugawara, K., Kazama, T., Watanabe, T.: Foraging behavior of interacting robots with virtual pheromone. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), pp. 3074–3079. IEEE Press, Piscataway (2004)
19. Werger, B., Matarić, M.: Robotic "food" chains: Externalization of state and program for minimal-agent foraging. In: Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats (SAB 1996), pp. 625–634. MIT Press, Cambridge (1996)
20. Wessnitzer, J., Melhuish, C.: Collective decision-making and behaviour transitions in distributed ad hoc wireless networks of mobile robots: Target-hunting. In: Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., Kim, J.T. (eds.) ECAL 2003. LNCS (LNAI), vol. 2801, pp. 893–902. Springer, Heidelberg (2003)

Positional Communication and Private Information in Honeybee Foraging Models

Peter Bailis[1], Radhika Nagpal[2,3], and Justin Werfel[3]

[1] Harvard College,
Harvard University, Cambridge, MA, USA
pbailis@eecs.harvard.edu
[2] School of Engineering and Applied Sciences,
Harvard University, Cambridge, MA, USA
rad@eecs.harvard.edu
[3] Wyss Institute for Biologically Inspired Engineering,
Harvard University, Boston, MA, USA
justin.werfel@wyss.harvard.edu

Abstract. Honeybees coordinate foraging efforts across vast areas through a complex system of advertising and recruitment. One mechanism for coordination is the waggle dance, a movement pattern which carries positional information about food sources. However, recent evidence suggests that recruited foragers may not use the dance's positional information to the degree that has traditionally been believed. We model bee colony foraging to investigate the value of sharing food source position information in different environments. We find that in several environments, relying solely on private information about previously encountered food sources is more efficient than sharing information. Relying on private information leads to a greater diversity of forage sites and can decrease over-harvesting of sources. This is beneficial in environments with small quantities of nectar per flower, but may be detrimental in nectar-rich environments. Efficiency depends on both the environment and a balance between exploiting high-quality food sources and oversubscribing them.

1 Introduction

Honeybee colonies are well-known for their ability to coordinate foraging over large areas and efficiently allocate labor among food sources. The predominant model for honeybee communication dictates that bees use a complex movement pattern known as the waggle dance to communicate positional information to unemployed foragers who then proceed to the indicated food source [10], [11]. However, it has recently been suggested that bees may instead rely primarily on private information and use publicly shared information only as a backup [8]. According to recent work, waggle-dancing may act primarily as a trigger that directs bees to forage previously known areas instead of following the dancer's positional cues about a food source.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 263–274, 2010.

We explore the role of private information in bee colony foraging and compare the relative efficiency of foraging in the traditional model of dance communication and in a model in which bees rely solely on their own internal positional information from past experience. We examine each model across several flower densities, distributions, qualities, and nectar quantities in order to determine when communicating positional information is advantageous.

We show that the benefit of sharing position information is highly dependent on the environment in which a colony operates and that relying solely on private information is more efficient than sharing information in several environments. Relying on private information results in a greater number of active forage sites. With small amounts of nectar per flower, this is beneficial as it decreases the risk of quickly exhausting food sources, resulting in fewer wasted foraging trips. Sharing positional information allows the colony to concentrate effort on foraging desirable and energy-efficient food sources at the risk of oversubscribing and quickly depleting them. With nectar-plentiful flowers, the risk of over-harvesting decreases and it becomes beneficial to concentrate foraging efforts.

In Section 2 we discuss related studies and previous models. In Section 3 we present our model and the two communication strategies we consider. Section 4 presents results and analysis. Section 5 concludes.

2 Related Work

Colony foraging behavior has been studied extensively, from bee foraging ranges [2] to the distribution of foragers and scouts [10], [16]. A system of advertisement and forager recruitment allows the colony to adjust forager allocation and respond to environmental changes [11]. Bees returning to the hive with food perform a waggle dance, a figure-eight vibration pattern carrying specific positional information [5], the duration of which is proportional to the source's quality [13]. The traditional understanding of the dance is that unemployed bees interpret it as explicit positional directions to a particular forage site [10].

Recent work has suggested that unemployed bees may not make full use of the position information contained in the waggle dance. An alternative hypothesis is that bees use the dance as a cue to return to previously discovered, privately known food sources [3], [8]. In one study, 93% of waggle dance recruits returned to internally remembered food sources rather than the source designated by dancers [7]. These findings are opposed to the traditional understanding of bee behavior, yet the extent to which bees rely on internal information and the relative benefits of sharing information are still in question [4], [14].

Prior work has studied the efficiency of the waggle dance in particular environments and communication models, but has not considered the potential role of private bee memory. Dornhaus et al. considered the role of recruitment and colony size within simulation and found that communication conferred the greatest benefit in worlds with few food sources arranged in scarce patches [6]. Beekman and Lew further explored the role of communication and found that communication allows colonies to efficiently exploit the most profitable food sources. The authors considered several communication models, including a model where recruits

searched for new food sources instead of following dance information [1], but recruits did not use any private information about previously known food sources. In our work, we focus on the recent discussions regarding the role of private information in bee forager allocation.

3 Model of Colony Foraging

We constructed an agent-based model that reflects current knowledge of bee foraging in nature.

3.1 World

We model a continuous world with discrete timesteps. The world is 12 km by 12 km with a hive at the center, corresponding to a 6 km radius of foraging activity [2], and bees cannot travel beyond its boundaries. Each timestep in our world corresponds to approximately 3.5 seconds of real time.

3.2 Flowers

Flowers represent food sources, and each has a predetermined quantity of nectar that can support a fixed number of forager trips before being exhausted. Nectar qualities are variable: flowers can contain either low-quality nectar (1 unit per trip) or high-quality nectar (4 units per trip), reflecting the proportions of sugar concentrations found in different flowers [6]. Because we model a short timeframe on the order of days, we do not simulate flower death.

We model two possibilities for flower distribution: evenly *scattered* and *clustered* (Fig. 1). In scattered worlds, food sources are randomly placed, similar to Dornhaus et al. [6]. In clustered worlds, food sources are distributed among a number of randomly placed clusters, and each flower is placed at a distance normally distributed ($\mu = 0$, σ =72 meters) from the cluster center, similar to Beekman and Lew [1]. Each cluster is of one particular quality, and we probabilistically add additional clusters at a predetermined rate in order to simulate a dynamic environment. We begin with six clusters in our world and end with approximately 10.5 clusters at the end of 100 hours.

Given a flower f distance D_f from the hive with nectar quality N_f and D_{max}, the maximum distance from the hive, we define the overall flower quality $Q(f)$:

$$Q(f) = \frac{D_{max} - D_f}{D_{max}} * N_f \tag{1}$$

This equation closely resembles a proposed metric for assessing flower desirability, (expected energy obtained - cost of trip)/(time of trip) [12], and prioritizes high-quality flowers closer to the hive, optimizing trip energy cost compared to the expected nectar yield. Under this model, a low-quality flower at distance n will be valued equivalently to a high-quality flower at distance $4n$.

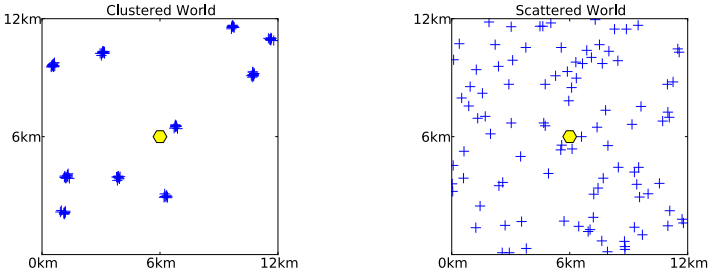


Fig. 1. Representative food distributions. Each world contains 100 flowers (+) and the hive (hexagon). Flowers are not drawn to scale.

3.3 Bees and Movement

We model foraging bees as simple agents. Bees can fly at a speed of 25 km/h in any direction and can detect and harvest nectar from flowers within a radius of 24 m [6]. Bees remember one flower location at a time and harvest nectar from one flower per trip from the hive. A bee can travel up to 6 km before it must return to the hive to replenish its energy stores. In the hive, bees deposit collected nectar.

We model the inaccuracies of bee flight directly instead of artificially causing flower location efforts to fail as in prior work [1] using two mechanisms: actuation error and perception error. At each timestep, the bee calculates its desired trajectory based on a goal position and the position at which it currently believes it is located. Because bee flight is imperfect, its actual trajectory is the desired trajectory plus a small perturbation, the *actuation error*. Sensory feedback lets the bee update its estimate of its position based on the actual rather than desired trajectory. However, sensing is imperfect, so the bee's position estimate is updated by the actual trajectory plus another perturbation, the *perception error*. For our simulations, we used independent perception and actuation errors randomly chosen between -3% and 3% per bee per timestep which were applied to the magnitude and angle of the movement vectors. To model familiarity with areas closer to the hive, when returning bees are within a short distance from the hive (120 m), their perceived position is updated to accurately reflect their actual position.

3.4 Bee Roles

Bees have one of three roles: *scout*, *forager*, or *unemployed*, similar to roles considered in related work [1], [6], [15].

A *scout* searches for new flowers by flying away from the hive and moving randomly throughout the world according to a Lévy flight pattern, a random walk with step length l distributed according to an inverse power law $P(l) \propto l^{-2}$ such that $l \in [24\text{ m}, 6\text{ km}]$ that is believed to closely approximate bee flight in nature [9] (Fig. 2). At each timestep, a scout surveys its surroundings and

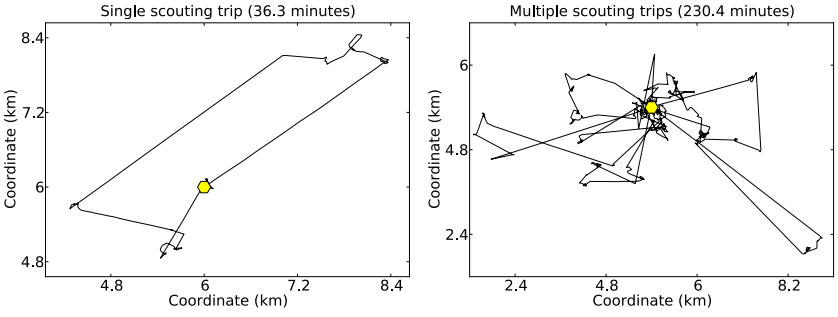


Fig. 2. Bee movement (Lévy flight) in single and multiple scouting trips from the hive

remembers the highest quality flower it has observed according to Equation 1. Upon depleting its energy, the scout returns to the hive, where it replenishes its energy supply and becomes a *forager*.

A *forager* is aware of a flower's position and repeatedly flies to it, collects nectar, and delivers the nectar to the hive. Because a bee's perceived position is not necessarily the same as its actual position, the coordinates it remembers may not reflect the flower's actual position in the world. When a forager perceives it has reached its remembered coordinates, it attempts to locate and harvest a flower. If none are in the vicinity whether due to positional error or flower depletion, the forager begins Lévy flight and searches for a new flower which it will then collect nectar from and subsequently remember according to its perceived position. Once the forager collects nectar or runs out of energy, it must return to the hive. Therefore, foragers who reach their known flower position but do not find a flower become similar to scouts, but upon finding nectar they collect it and return (Fig. 3).

An *unemployed* forager has no known flower and must either wait to be recruited or become a scout. At each timestep, with a small probability (.1%), it becomes a scout and searches for a food source [3], [15].

3.5 Flower Quality, Foraging, and Recruitment

Upon returning to the hive with a known flower location, a bee has three choices: forget about the known flower, continue to forage from the flower, or continue to forage from the flower after attempting to recruit additional foragers.

In order to adapt to changing environments and different conditions, bees maintain dynamic thresholds that determine whether a bee should remember or forget about a current food source as well as whether a bee should advertise the flower using the waggle dance [11], [12]. While the threshold for abandoning a source is a function of the individual flower, the threshold for dancing is complicated and depends on colony-wide metrics like the ability to find a food-storer bee [15]; because we do not model these features of the colony, we use individually-tuned threshold mechanisms for both which quickly converge at

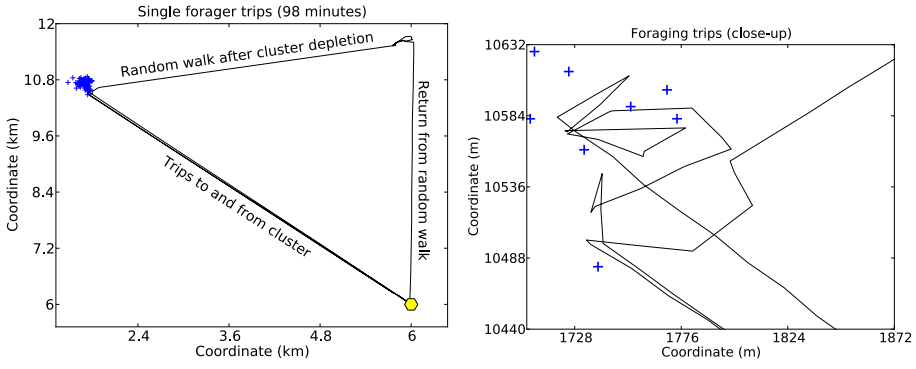


Fig. 3. Movement of a single bee making foraging trips to and from a cluster of flowers. After one successful trip, the bee returns to the patch and, because the nearby flowers are depleted (due to other bees foraging as well), the forager begins Lévy flight in search of a new flower (traveling to the right corner), eventually returning to the hive. Note that even though the forager believes it has returned to the same position each trip, the actual positions at which it arrives are slightly different due to movement error, as seen in the close-up.

the colony level in practice. Each bee has a *harvest threshold*, t_h, and a *dance threshold*, t_d, with the invariant that $t_h \leq t_d$. Each time a bee returns to the hive with a known flower, it assesses the quality relative to its thresholds in order to determine the appropriate action. The chosen action also affects the thresholds, creating a dynamic feedback mechanism.

Given a flower f with quality $q = Q(f)$, we have three possibilities:

$q < t_h$: The flower quality is poor relative to the flowers the bee has recently encountered. The bee becomes unemployed, but lowers its standards for foraging:

$$t_h \leftarrow t_h - \frac{t_h - q}{2} \tag{2}$$

$t_h < q < t_d$: The flower quality is better than others that the bee has recently encountered, but not of sufficient quality to merit advertising to other bees. The bee remembers the flower and continues to harvest it. The bee raises its standards for harvesting and lowers its standards for advertising.

$$t_d \leftarrow t_d - max(.1, \frac{t_d - q}{2}), t_h \leftarrow min(t_h + max(.1, \frac{q - t_h}{2}), t_d) \tag{3}$$

$q > t_d$: The flower quality is higher than the flowers that the bee has recently encountered. The bee remembers the flower, advertises it to other bees, and continues to harvest it. The bee raises its standards for advertising:

$$t_d \leftarrow t_d + \frac{q - t_d}{2} \tag{4}$$

In order to model a scout's willingness to find new food sources, scouts halve both of their thresholds upon leaving the hive.

3.6 Hypothetical Communication Models

Bees advertise flowers by "dancing" inside the colony. Given a flower f of quality $q = Q(f)$, if a bee decides to advertise f, it will do so for $c * q$ timesteps. In our model, $c = 5$. A bee can recruit exactly one other bee per timestep as only a limited number of bees can physically observe a waggle dance at a given time. Recruits are chosen randomly from the pool of unemployed bees [12].

We consider two possible communication models for recruitment:

- **Model SharePosition**. In this traditional position-sharing model, recruits become active foragers at the advertised position. Because the flower coordinates are specified according to the dancer's perceived location from when it found the flower, the recruit may not find it due to actuation and perception errors of both the dancer and the recruit or because of flower depletion.

- **Model PrivatePosition**. In this private-information model, recruits ignore the positional information a dancer provides and begin to forage at their previously known flower position, or, if they have none, they become scouts. This model deemphasizes the role of positional information in foraging coordination and instead focuses on private information [8]. In this model, dancing reactivates unemployed foragers instead of providing direction regarding where to forage.

These two models represent the accepted understanding of bee communication as well as new theories about bee communication, respectively.

4 Results

We evaluated the relative efficiency of the communication models in several distinct environments within a custom simulator, varying flower generation rates, flower distributions, and flower nectar quantities (Fig. 4). For each configuration, we simulated 96 hours of foraging with a colony of 500 bees. We do not consider different colony sizes, but consistent with prior work [6], colony size did not significantly impact our results.

Relative foraging efficiency (nectar gathered per unit of energy expended) depended greatly on the environment; in several cases PrivatePosition outperformed SharePosition (up to 40%), but, in others, SharePosition was more efficient (up to 35%). At low flower generation rates, PrivatePosition performed better in scattered worlds than in clustered worlds, but, at high flower generation rates, it performed better in clustered worlds. PrivatePosition's relative performance was better in homogeneous worlds than in heterogeneous worlds. In general, as nectar quantity increased such that a given flower could support more repeated trips, the relative efficiency of PrivatePosition decreased. Similarly, as flower generation rates increased, PrivatePosition became less effective.

The food distribution influenced the usefulness of information sharing. In scattered worlds, bees were likely to find flowers independently, especially at

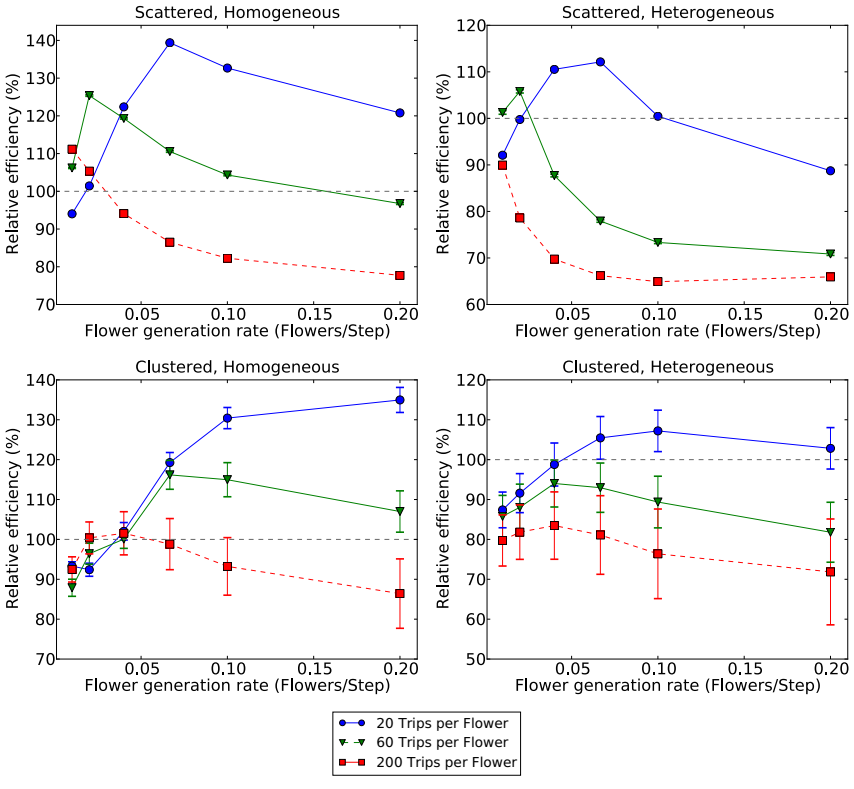


Fig. 4. Relative efficiency of PrivatePosition compared to SharePosition. Efficiency is defined as the ratio of nectar retrieved to energy expended. Each data point represents 100 trials in separate, randomly-generated environments matching the specified configuration. Error bars show the standard error on the mean.

moderate to high flower generation rates; sharing did not confer a serious advantage because flower locations were not geographically correlated. In clustered worlds, however, once a bee found a cluster, it effectively found many flowers, and sharing information allowed bees to find clumps faster than if they searched independently. With heterogeneous nectar qualities, however, sharing information about food sources became more valuable as coordination allowed the colony to concentrate on both nearby flowers and higher quality nectar.

PrivatePosition was aware of a greater number of food sources on average than SharePosition, which had several consequences for efficiency. We examined the average number of food sources the colony exploited at any given time (Fig. 5), and found that PrivatePosition foraged up to 114% more flowers in scattered worlds and 25% more clusters in clustered worlds. As the number of known food sources decreased, each was visited by a greater number of foragers and was depleted more quickly. Once depletion occurred, foragers needed to find a

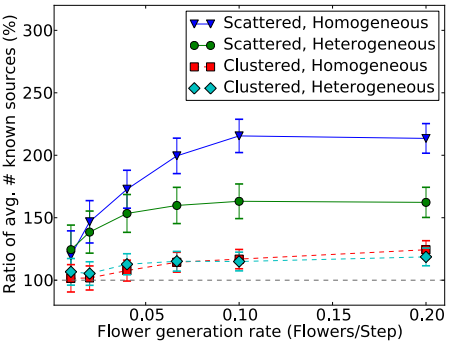


Fig. 5. Average number of clusters or flowers known to actively foraging bees at each timestep, expressed as a ratio of PrivatePosition to SharePosition with flower nectar quantity limited to 20 trips. If a bee knew of a food source, had visited it, and the source was not depleted, then the colony was considered aware of the food source. The clustered worlds contained 10 randomly placed flower clusters and we counted a flower as belonging to a cluster if the flower was within 480 m of the cluster center. If a flower was located within multiple cluster radii, we counted the flower as belonging to each cluster. Each data point represents the average of 30 trials, and error bars show the standard error on the mean.

replacement flower, which, depending on the environment, could be an energy-intensive process. In scattered worlds, adding additional flowers made it easier to find a replacement. In clustered worlds, however, adding more flowers only increased the amount of food at existing cluster locations and did not contribute additional geographic diversity to the food distribution. This partly explains the noise in the cluster scenarios: placing flowers at a cluster had similar effects to placing a single large, high-capacity flower. As long as some flowers remained in a cluster, foragers could likely find a replacement, but, if a cluster was entirely depleted, finding another food source nearby was unlikely. Because flowers were constantly generated, subsequent returns to a depleted cluster would possibly yield some nectar, which is advantageous for PrivatePosition, however at low rates of generation this resulted in many wasted trips.

SharePosition favored foraging from better food sources and quickly depleting them, while PrivatePosition favored foraging from many possibly inferior sources and had more successful forager trips. We examined the number of foraging trips that did not result in nectar harvesting (Fig. 6), and SharePosition consistently had a higher proportion of unsuccessful trips. SharePosition was more effective with high flower nectar quantities than with low flower nectar quantities, but was still less successful overall than PrivatePosition. In worlds with few flowers, the two models were almost equivalent because food sources were hard to find and easy to deplete. Even though recruits in SharePosition received position information, they were less successful than in PrivatePosition; SharePosition foragers did not find their food sources as often as PrivatePosition foragers did.

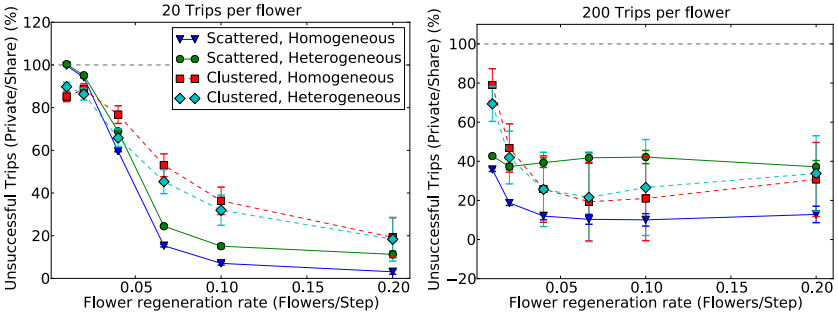


Fig. 6. Unsuccessful foraging trips as a fraction of the total number of foraging trips in PrivatePosition compared to SharePosition across multiple environments. A trip was unsuccessful if a forager returns to the hive without food. Each data point represents 100 different simulated worlds, and error bars show the standard error on the mean.

PrivatePosition's relative foraging success was due to SharePosition's oversubscription of the more favorable food sources. At low flower nectar quantities, SharePosition's concentration of foraging efforts resulted in rapid source depletion, wasted trips, and difficulty finding new flowers. In scattered, homogeneous worlds, for example, in PrivatePosition, the colony harvested from a wide range of food sources, while in SharePosition, the colony harvested from food sources only within a close radius of the hive, decreasing flight time but increasing its proportion of unsuccessful trips. Increasing flower nectar quantity reduced the risk of oversubscribing any particular food source and supported greater forager concentration. With higher nectar quantities, SharePosition still had fewer successful trips, but each successful trip was more valuable (due to trip speed or nectar quality), resulting in a higher overall efficiency (Fig. 4). Similarly, with one trip-worth of nectar per flower, there would be no benefit to sharing information (except due to geographic flower locality), but with infinite nectar per flower, sharing information would greatly benefit the colony.

The colony must balance the tradeoff between focusing on fewer, more desirable food sources and oversubscribing them. While sharing information about flowers in SharePosition resulted in higher nectar quantities, if a food source was quickly depleted by foragers, the benefit of sharing information decreased due to wasted energy and forager effort. Efficiency largely depended on whether, by recruiting additional foragers to a food source, the colony wasted energy upon source depletion that could have been avoided by spreading out the foragers among other sources instead. As flowers supported more foragers, the colony could concentrate on better food sources without the risk of quick depletion, and the benefit of sharing information increased.

We summarize several of the important factors in determining model success:

- When the nectar quantity per food source increases, the importance of harvesting from a variety of food sources decreases as depletion slows. The benefit of sharing information about high-quality food sources increases.
- Exploiting a variety of food sources is useful when they are easily depleted, but, as food sources become easier to find, oversubscribing is less harmful as foragers are more likely to find replacement flowers.
- In heterogeneous environments, exploiting the most profitable food sources may outweigh the cost of over-harvesting, so position sharing is more beneficial. In homogeneous worlds, this effect is lessened as the difference between the best and worst sources is smaller.
- In environments with many flowers located closely together (as in clustered worlds), the importance of exploiting a variety of food sources decreases compared to environments with scattered food sources; it is easier to find another flower nearby once a targeted flower becomes depleted. However, once a cluster is depleted, it may be difficult to find a new food source.

5 Conclusion

In evaluating the benefit of communicating positional information in bee foraging, one must consider a variety of factors: flower quality, quantity, capacity, and distribution. Our results show that, under appropriate conditions, relying on internal information can be more efficient than sharing information. This supports recent studies about private information that run contrary to the traditional interpretation of information sharing in honeybees. It is plausible that this behavior occurs naturally as a response to particular environment types, particularly when food source diversity is important.

In this work, we have provided evidence for the efficiency of relying on private information within a bee foraging model in several environments and explained factors for each model's success. Our results have consequences not only for biologists but for system designers who are faced with decisions about communication and task allocation in a particular environment. Depending on environmental characteristics, the additional costs of incorporating a communication system may outweigh the benefit of doing so if one can instead rely on internal agent memory. While field biologists have not yet conclusively determined the importance of sharing positional information in honeybee foraging, we have shown that in some environments relying solely on private information may be more efficient than sharing position information.

Source code and Traces. All source code and traces are available at `http://eecs.harvard.edu/~pbailis/beesim/`.

Acknowledgements. The authors would like to thank Peter Lifland for his assistance with early simulation and modeling. This work was supported in part by NSF Expeditions Grant IIS-926148 and the Wyss Institute for Biologically Inspired Engineering.

References

1. Beekman, M., Lew, J.B.: Foraging in honeybees–when does it pay to dance? Behavioral Ecology 19(2), 255–261 (2008)
2. Beekman, M., Ratnieks, F.L.W.: Long-range foraging by the honey-bee. Apis mellifera L. Functional Ecology 14, 490–496 (2000)
3. Biesmeijer, J.C., Seeley, T.: The use of waggle dance information by honey bees throughout their foraging careers. Behavioral Ecology and Sociobiology 59, 133–142 (2005)
4. Brockmann, A., Sen Sarma, M.: Honeybee dance language: is it overrated? Trends in Ecology and Evolution 24, 583 (2009)
5. Dornhaus, A., Chittka, L.: Why do honey bees dance? Behavioral Ecology and Sociobiology 55, 395–401 (2004)
6. Dornhaus, A., Klügl, F., Oechslein, C., Puppe, F., Chittka, L.: Benefits of recruitment in honey bees: effects of ecology and colony size in an individual-based model. Behavioral Ecology 17(3), 336–344 (2006)
7. Grüter, C., Balbuena, M., Farina, M.: Information conflicts created by the waggle dance. Proceedings of the Royal Society Biological Sciences 275, 1327 (2008)
8. Grüter, C., Farina, W.: The honeybee waggle dance: can we follow the steps? Trends in Ecology and Evolution 24, 242–247 (2009)
9. Reynolds, A.: Cooperative random Lévy flight searches and the flight patterns of honeybees. Physics letters A 354, 384–388 (2006)
10. Seeley, T.: Division of labor between scouts and recruits in honeybee foraging. Behavioral Ecology and Sociobiology 12, 253–259 (1983)
11. Seeley, T.: Honey bee foragers as sensory units of their colonies. Behavioral Ecology and Sociobiology 34, 51–62 (1994)
12. Seeley, T.: The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies. Harvard University Press, Cambridge (1996)
13. Seeley, T., Mikheyev, A.: Dancing bees tune both duration and rate of waggle-run production in relation to nectar-source profitability. Journal of Comparative Physiology A 186, 813–819 (2000)
14. Shermin, G., Visscher, P.: Honeybee colonies achieve fitness through dancing. Nature 419, 920–922 (2002)
15. de Vries, H., Biesmeijer, J.C.: Modelling collective foraging by means of individual behaviour rules in honey-bees. Behavioral Ecology and Sociobiology 44, 109–124 (1998)
16. Waddington, K., Holden, L.: Optimal foraging: on flower selection by bees. The American Naturalist 114 (1979)

Rank Based Particle Swarm Optimization

Affan Khan[1], Muhammad Sadeequllah[2],
Riaz-ul-Hasnain[3], and Azzam-ul-Asar[3]

[1] Dept of Computer Science, University of Peshawar, N.W.F.P, Pakistan
[2] Institute of Information Technology,
Kohat University of Science & Technology, N.W.F.P, Pakistan
[3] Dept of Electrical & Electronics Engineering,
University of Engineering & Technology, Peshawar, N.W.F.P, Pakistan
{khanaffan,sadeeqkhan,sriazh}@yahoo.com

Abstract. Population members of the classical PSO quickly converge onto a smaller region of the objective function landscape, which helps to refine the discovered optimum, but the searching ability of the algorithm collapses. This paper proposes modification to the way information flows between the global best particle and the rest of the particles to resist particles clustering. Particles are ranked according to their fitness value such that each rank is single particle and a particle learns only from the particle one rank above. Global best particle learns only from its own experience. The proposed version of PSO is named as RPSO and experiments on test bed functions show not only RPSO particles resisted clustering but stability has also been observed in RPSO results. The downside of RPSO was its slow rate of convergence, which was improved by nominating certain particles from the whole population as diggers with learning topology of the classical PSO. This version was named RPSO-D and experiments were conducted to show its superiority over the classical PSO, both in terms of stability and rate of convergence.

1 Introduction

Stochastic component is used in stochastic algorithms to introduce and maintain diversity, which helps these algorithms to maintain global search ability. If an algorithm heavily relies for its results on stochastic component, high variations could be found in its outcomes for the same problem. It is highly desirable that a stochastic optimization algorithm should have both high rate of convergence and it should consistently find good results. This latter property is also known as stability of the algorithm, and is rarely given any attention by the research community. Particle Swarm Optimization PSO [1,2] is one such population based stochastic algorithm in which much has been done about convergence speed, but very little is found in the literature about stability. Some facts which are known about PSO and contribute to instability in PSO results are the followings.

1. Relative distances between particles tend to decrease over time and particles start clustering.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 275–286, 2010.

2. Since there is zero clustering at the time of initialization, population diversity is at maximum and its global searching capability is the highest, which decays with time as soon as cluster forming begins.
3. When particles cluster, PSO turns into excellent digger. If all particles are in the same basin, PSO starts behaving like local search, its global diversity and search capability is lost. At the best, even if they get out of the current basin, they will be too slow to find another basin of a better optimum.
4. In contrary to evolutionary algorithms, where members are allowed to fly in any direction, PSO particles are allowed to fly only in those directions which are, from experience, considered good. Hence, PSO has better digging capability at the cost of global search ability.
5. A phenomenon, known as stagnation [3], can occur when all particles catch up position of the global best particle. At this point, all particles stop moving and algorithm shows no improvement. This can be premature convergence as it does not even guarantee that algorithm has converged on a local minimum.

What contributes to these features of PSO is the clustering. Clustering is formed due to the way information flows in PSO [4]. Best trend prevails quickly and hostages the whole swarm to follow it. This makes PSO unstable and highly dependent upon initial positions. In order to maintain a modest diversity, If information dissemination is decelerated, it will not only make PSO stable but it will also improve its searching ability.

Clustering process in PSO is relatively slow in lbest model than gbest. In gbest model, all particles learn from a single particle, called the global best, and hence, a single center of gravity and more quick is the clustering. In lbest model, a particle learns from the best particle of its group, called the local best. Hence, more the groups more will be the gravity centers, high population diversity and slow clustering. This is because information flow from the global best to the global worst particle is slow. It has also been observed that gbest model posses high rate of convergence than lbest and is more suitable on uni-modal functions [10]. While lbest results in better optima on multi-modal functions.

To retain population diversity, researchers have experimented with lbest model. Even though population partitioning in original lbest is pure index based, a different partitioning scheme based on spatial location of particles was proposed by Suganthan [5]. During each iteration of the algorithm, a distance from each particle to every other particle in the swarm is computed and the largest distance between any two points is denoted by d_{max} . A particle neighborhood is decided by computing $\left\| \frac{x_{cur} - x_a}{d_{max}} \right\|$ for each particle, where x_{cur} is the current particle and x_a is any other particle. x_{cur} and x_a are considered neighbors if the calculated distance ratio is below some threshold value. This threshold value is very small in the start (i.e. lbest model) to encourage large number of groups to search different areas of the search space. Threshold value is gradually increased to 1 (i.e. become gbest) to leverage digging capability of the PSO.

In a different effort, Kennedy experimented with lbest model by defining different topologies based on connections among particles [6]. Three topologies named ring, wheel and star were considered. Ring topology is the original lbest

with $l = 1$, where l is the neighborhood size, and with varying number of randomly interchanged connections. In wheel topology, all particles are connected with a single hub particle and no direct connection between them. Star topology is the original gbest. Since information spread in ring and wheel topologies was slower than star topology, these algorithms were robust in face of multiple local minima. Star topology, due to faster spread of information, showed high rate of convergence on unimodal functions.

Based on human social interaction studies, which indicate that people follow collective belief of a group rather than individuals, Kennedy presented another lbest version, which is mix of both ring topology and spatial neighborhood, called social stereotyping [7]. K-means clustering algorithm was used to assign clusters to particles, provided number of clusters were decided beforehand. Besides clustering, neighborhood was decided for particles based on the original ring topology. Centroid of the cluster containing particle i is denoted by $\overline{C(i)}$ and computed according to (1).

$$\overline{C(i)} = \frac{1}{|C_l|} \sum_{a \in l} a \ . \tag{1}$$

Where, $|C_l|$ is the number of elements in cluster l. Neighborhood best particle, denoted by y_g, is selected according to (2).

$$y_g \in N_i \,|\, f(y_g) \leq f(y_b) \ \forall\, y_b \in N_i \ . \tag{2}$$

Centroid of the cluster that contain neighborhood best particle g is denoted by $|\overline{C(g)}|$. Based on these definitions the following three variants of lbest velocity update equation were experimented.

$$v_{i,j} = wv_{i,j} + c_1 r_{1,j} \left(\overline{C(i)_j} - x_{i,j} \right) + c_2 r_{2,j} \left(\hat{y}_{i,j} - x_{i,j} \right) \ . \tag{3}$$

$$v_{i,j} = wv_{i,j} + c_1 r_{1,j} \left(y_{i,j} - x_{i,j} \right) + c_2 r_{2,j} \left(\overline{C(g)_j} - x_{i,j} \right) \ . \tag{4}$$

$$v_{i,j} = wv_{i,j} + c_1 r_{1,j} \left(\overline{C(i)_j} - x_{i,j} \right) + c_2 r_{2,j} \left(\overline{C(g)_j} - x_{i,j} \right) \ . \tag{5}$$

Even though, because of clustering overheads, these algorithms were slower than original lbest, Kennedy reported that (3) performs better that original lbest on some problems. Equation (4) and (5) performed worse, suggesting that particles should not emulate centroids of remote clusters.

The idea of subpopulation has also been applied in PSO [8], which was originally applied in evolutionary algorithms, e.g. GA [9]. The idea is to use multiple distinct subpopulations searching different regions of the search space and, from time to time, exchange members between subpopulations. In PSO, each subpopulation has its own global best particle and arithmetic crossover operator was applied. When selecting parents for crossover, there is a small probability that one of the parents might be selected from a different subpopulation. Results regarding the method indicate that no performance improvement was recorded.

All these algorithms were intended to maintain a modest diversity in PSO population, but these methods either show insignificant improvement or improvement shown, if any, does not worth the overheads incurred. Only ring topology is prominent in that it shows good results with almost no overheads. Even in the ring topology, information propagation from the global best to the whole swarm is faster because information spreads both in clockwise as well as anticlockwise directions. This paper proposes a new algorithm, which could be called string topology, in which information propagation from the global best to the global worst is the slowest. Based on the fitness value of particles, the notion of single particle ranks is used. All particles are sorted according to their fitness, with global best on the first rank and global worst on the last. Each particle learns only from the particle one rank above, while global best learns only from its own experience. This approach is named RPSO and experiments indicate that it is highly stable in face of both single and multiple local minima. To bring better digging capability of gbest model in RPSO, a variant named RPSO-D is also presented where a fraction of the swarm is nominated as diggers which follow the learning topology of gbest model.

This paper is organized in the following manner. Section 2 gives background knowledge on PSO. In sect. 3, the proposed algorithm is presented. To show strengths of the proposed algorithm, results are computed in sect. 4.

2 Particle Swarm Optimization (PSO)

Proposed by Kennedy and Eberhart [1,2], Particle Swarm Optimization (PSO) is a population based stochastic optimization algorithm inspired from the social behavior of certain organisms, e.g. flock of birds and schooling of fish. Each member of the population is a potential solution, which flies with velocity v in $n-dimensional$ hyperspace. If s is the size of the swarm, then for each particle i: x_i is the current position of the particle, v_i is the current velocity of the particle and y_i is the personal best position of the particle. The personal best position y_i is the position that this particle has visited which yielded the high fitness value. In minimization context, a position yielding smaller function value is regarded high fitness. Update rule for y_i is given by (6).

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) & \text{if } f(x_i(t+1)) < f(y_i(t)) \end{cases} . \tag{6}$$

The particles follow three principles, as described by Kennedy:

i. Evaluating:- learning through self experience
ii. Comparing:-learning through comparative study
iii. Imitating:- learning through adapting the best trend

Each particle makes its own decisions and is influenced by its neighbors as well. As soon as a particle finds the best solution, it starts influencing its neighbors. This process converges all particles to an optimal point similar to cultural trends in human society. Interaction with the neighborhood is represented by $\hat{y}$, which

is the best position discovered by any particle in the neighborhood. Velocity and position update equations for j^{th} dimension of any particle i, at iteration $t+1$, are represented by (7) and (8) respectively.

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) + c_2 r_{2,j}(t)(\hat{y}_{i,j}(t) - x_{i,j}(t)) \ . \quad (7)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \ . \quad (8)$$

$r_1 r_2 \in \mathrm{U}(0,1)$ are independent random sequences which are scaled by constant $0 < c_1, c_2 \leq 2$. These constant are known as acceleration coefficients, and they constraint the maximum distance covered by any particle in a single iteration. The original PSO algorithm is written in Fig. 1.

Based on the neighborhood size, from which $\hat{y}$ is selected, two models of PSO gbest and lbest are used. In gbest, a single neighborhood is defined, which is the whole swarm. $\hat{y}$ is called the global best as it is the best position discovered by any particle in the whole swarm.

$$\hat{y}(t) \in \{y_1(t), y_2(t), \cdots, y_s(t)\} \ | f(\hat{y}(t)) = \min\{f(y_1(t)), f(y_2(t)), \cdots, f(y_s(t))\} \ . \quad (9)$$

In lbest model, overlapping neighborhoods of radius l are defined and $\hat{y}$ is called the local best or neighborhood best. There is no relationship among the particles in the same neighborhood, the selection is pure index based which wraps around the population size s. Equation (10) and (11) define i^{th} neighborhood of size l, denoted by N_i, and local best particle $\hat{y}$ respectively.

$$N_i = \{y_{i-l}(t), y_{i-l+1}(t), \cdots, y_{i-1}(t), y_i(t), y_{i+1}(t), \cdots, y_{i+l-1}(t), y_{i+l}(t)\} \ . \quad (10)$$

$$\hat{y}(t+1) \in N_i | f(\hat{y}_i(t+1)) = \min\{f(a)\}, \forall a \in N_i \ . \quad (11)$$

lbest models maintains multiple gravity centers, which helps PSO to avoid premature convergence and the algorithm become much robust in face of multiple optima. However, gbest model shows faster convergence on unimodal functions. In lbest, when $l = s$, lbest becomes gbest.

```
Create and initialize an n-dimensional PSO: S
repeat:
        for each particle i ∈ [1, ···, s]:
                if f(S.x_i) < f(S.y_i)
                        then S.y_i = S.x_i
                if f(S.y_i) < f(S.ŷ)
                        then S.ŷ = S.y_i
        end for
        Perform PSO updates on S using (7)( 8)
until stopping condition is true
```

Fig. 1. PSO algorithm

3 The Proposed Particle Swarm Optimizer

3.1 Rank Based Particle Swarm Optimizer (RPSO)

To bring stability in PSO, information sharing among particles is strictly restricted by introducing the concept of ranks. According to their fitness, all particles are arranged in a line such that particle with best fitness value is at the front and particle with least fitness value is at the tail. Each particle represents a unique rank with best particle having rank 1 and least best at n^{th} rank, assuming n is the swarm size. Any particle with rank r learns only from the particle with rank $r-1$, and best fit particle learns only from its own experiences (i.e. its social learning is disabled and only its cognitive component is utilized). PSO with this topology has been named here as Rank based Particle Swarm Optimizer (RPSO) and is graphically illustrated in Fig. 2.

The topology of RPSO halts the noise to propagate through the swarm, as attraction from a suboptimal might affect the leading particles quickly, but its impact will reduce as it reaches the tail. This is because of the fact that information exchange among particles has been decelerated by traveling down from particle to particle to reach the tail. This helps PSO to maintain a modest diversity longer than the classical PSO, and could be observed in Fig. 3.

Another phenomenon seen in gbest PSO is frequent switching of the global best particle, which speeds up as particles start clustering. This forces swarm to keep changing directions and the effective work done by particles is negligible. The topology of RPSO also remedies this problem such that only few particles lead the swarm. This fact could be better understood in Fig. 4, which clearly shows that, in conventional PSO, the number of times a particle has been selected as global best is somehow uniform. On other hand, in RPSO, only few particles have very high frequency of being selected as global best while most of the particles have never been selected at all. RPSO update equation and its algorithm is given by (12) and Fig. 5 respectively.

Even though, as shown by results in this paper, ranks based PSO preserves population diversity and provides stable convergence, however, when the swarm lands into basin of the optimum, where strong local search is required, RPSO becomes much slower than conventional PSO. Slow digging ability near the optimum makes

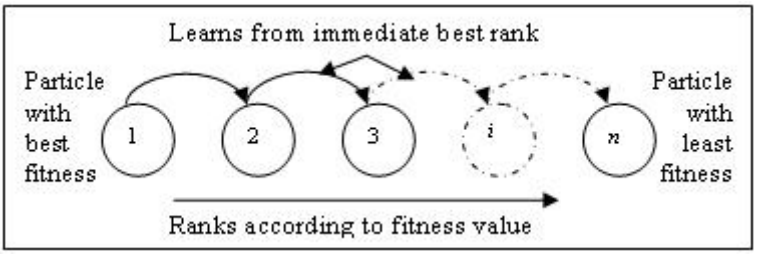


Fig. 2. Information flow in RPSO

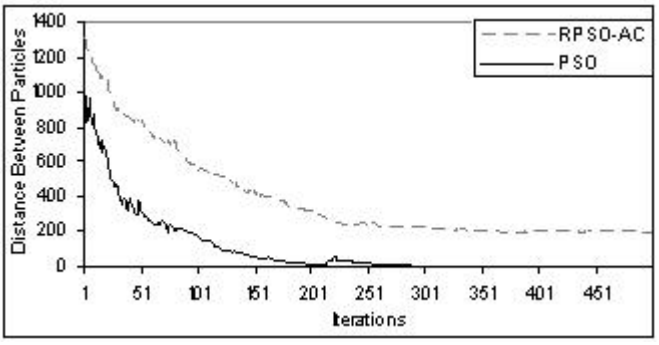


Fig. 3. Diversity in population—Rastrigin

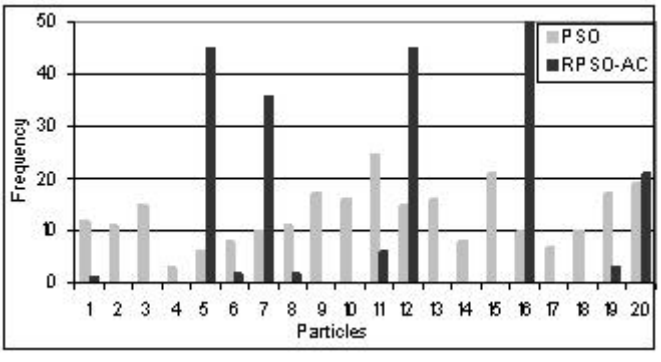


Fig. 4. Frequency of each particle as Global Best

RPSO impractical on functions with deep local optima. This problem could be alleviated if some particles in the swarm are assigned the duty of digging, which is discussed in the next subsection.

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)\{y_{i,j}(t) - x_{i,j}(t)\} + c_2 r_{2,j}(t)\{\hat{y}_{r-1}(t) - x_{i,j}(t)\} \quad (12)$$

3.2 RPSO with Diggers (RPSO-D)

PSO turns into excellent digger when it dives into the basin of optimum. RPSO lacks this ability because it resists clustering. To bring digging capability of the conventional PSO into RPSO, a small number of particles, denoted by λ, are declared as diggers. Diggers follow the topology of gbest model and learn from the global best particle. If p is the swarm size then $0 \leq \lambda \leq p$, and the remaining $p - \lambda$ particles, if $\lambda < p$, follow RPSO topology. RPSO with diggers has been named here as RPSO-D. With $\lambda = p$, RPSO-D becomes the classical gbest PSO and with $\lambda = 0$, RPSO-D become RPSO. Figure 6 is the graphical illustration and Fig. 7 is the algorithm of RPSO-D.

```
Create and initialize n-dimensional PSO: S
Create and initialize rank list l consisting all particles
repeat:
    for each particle p ∈ [1, ··· , s]:
        if f(p.x) < f(p.y)
            then p.y = p.x
        Perform updates on p using (12)( 8)
        update-rank(l, p)
    end for
until stopping condition is true
```

```
update-rank(l, p)
for each i ∈ l
  if i <> p and f(i) > f(p)
      then insert(p, i, l)
end for

insert function will insert p
before i in list l
```

Fig. 5. RPSO algorithm

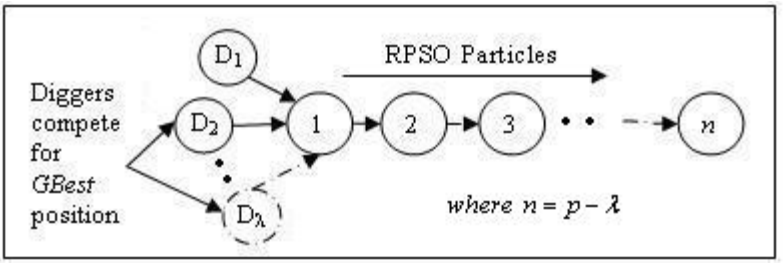


Fig. 6. Information flow in RPSO-D

```
Create and initialize an n-dimensional PSO: S
Create and initialize rank list l containing ranked particles
Label remaining particle in S as diggers
repeat:
        for each particle p ∈ [1, ··· , s]:
                if f(p.x) < f(p.y)
                        then p.y = p.x
                if is-digger(p) = true
                        Perform update on p using (7)(8)
                else
                        Perform update on p using (12)(8)
        end for
until stopping condition is true
```

Fig. 7. RPSO-D algorithm

Empirical analysis of diggers reveals that the overall behavior of RPSO-D is stable and it resists clustering, but its convergence rate is slow. On the other hand, PSO is unstable and rely heavily on the initial positions of particles in the search space. If number of diggers are increased in RPSO-D, it impedes convergence, but instability also creeps-in back. Hence, it is important to decide what ratio of diggers in population will be a better compromise.

Table 1. Diggers comparison in RPSO-D

Function	λ	Min	Max	Avg	Variance
Sphere	5	0	1.17E-236	3.38E-238	0
	3	1.47E-225	3.81E-212	7.84E-214	0
	0	9.47E-278	1.72E-175	2.54E-177	0
Rosenbrock	5	1.357E-21	5.5551705	0.7233408	2.3244047
	3	3.57E-15	1.66E+01	9.06E-01	5.0017239
	0	1.37E-07	13.90299	0.731362	3.8809401
Rastrigin	5	0.9949591	30.84366	8.825905	34.657233
	3	1.9899181	37.092145	9.0990844	35.905747
	0	0.9949591	32.580239	9.6545678	41.686936
Griewank	5	0	0.05176	0.0049733	8.406E-05
	3	0.00E+00	2.71E-02	4.82E-03	4.931E-05
	0	0	0.0221857	0.0033577	3.514E-05

Table 1 presents results of RPSO-D applied on different synthetic library functions. Population size is 20 particles, each one of 20 dimensions, and 10 kilo iterations has been performed, in which minimum, maximum, average and variance in error have been displayed. For a test function, error is the difference between the final fitness value obtained and its global minimum. The λ column represents number of diggers in the population. Results for Rosenbrock indicate that error obtained is the smallest for 5 diggers, then for 3 and lastly for 0 diggers. This shows that more diggers has found better optimum but at the same time it has also increased variance in the errors obtained. On Rastrigin function, it has also found better minimum but, in the same way, at the cost of increased variance, even though its average error is still low.

The above results indicate that large value of λ helps in finding better optima but it also bring instability in behavior of RPSO-D. Small value of λ keeps RPSO-D more consistent but more iterations are require to find a better optima. Moreover, at small value of λ, RPSO-D resists clustering and is less susceptible to be traped in local minima.

Dynamic Value of *lambda*. It is obvious from the above discussion that, in start of the algorithm, more searching is required and λ should be very small to allow RPSO-D to maintain diversity and avoid clustering. When algorithm reaches into the basin of a better optimum then digging capability is required and λ should be increased to support it. Hence, it is desirable to start RPSO-D with small λ and keep it increasing as the algorithm converges onto an optimum. Equation (13) models a linear relationship between λ and the number of iterations.

$$\lambda = population \times \frac{i}{max\ epoch} \quad . \tag{13}$$

In (13), i represents the i^{th} iteration, *maxepoch* is the total number of iterations and *population* is the population size.

4 Experimental Results

This section compares RPSO and RPSO-D for stability and anti-clustering capabilities against the classical PSO model. This comparison also applies to all variants of PSO which preserve information dissemination topology of PSO. The following functions have been used as test functions.

Spherical: It is a simple uni-modal function with its global minimum located at $x^* = 0$, with $f(x^*) = 0$. This function has no interaction between its variables.

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \tag{14}$$

Rosenbrock: A uni-modal function, with significant interaction between some of the variables. Its global minimum is located at $x^* = (1, 1, \cdots, 1)$, $f(x^*) = 0$.

$$f_2(x) = \sum_{i=1}^{\frac{n}{2}} (100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2) \tag{15}$$

Griewank:A multi-modal function with significant interaction between its variables caused by the product term. The global minimiser, $x^* = 0$, yields a function value of $f(x^*) = 0$.

$$f_3(x) = \frac{1}{400} \sum_{i=1} n x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{16}$$

Rastrigin:A multi-modal version of the spherical function, characterized by deep local minima arranged as sinusoidal bumps. The global minimum is $f(x^*) = 0$, where $x^* = 0$. The variables of this function are independent.

$$f_4(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10) \tag{17}$$

Experiments have been conducted for 10, 20, 30 and 40 dimensions, population $size = 20$, $c1, c2 = 1.49618$, $k = 0.729844$, $x \in [-100, 100]$, $errorthreshold = 0$, $epochs = 10,000$ and velocity clipping applied. Variance has been calculated for PSO, RPSO and RPSO-D with 3 diggers. To test better the empirical behavior of the algorithm against the above configuration, each value presented in both Table 2 and Table 3 is the average 100 runs. Total variance shown in these tables is the aggregate variance of a specific algorithm for all of the four test functions and on all four dimensions. Improvement against PSO is calculated using (18) and (19) for RPSO and RPSO-D respectively. Table 2 shows that, in comparison with PSO, variance in RPSO has been reduced to 84 percent. Table 3 shows improvement in variance of RPSO-D with respect to PSO is 92 percent.

$$\frac{PSO - RPSO}{PSO + RPSO} \times 100 \tag{18}$$

$$\frac{PSO - RPSOD}{PSO + RPSOD} \times 100 \tag{19}$$

Table 2. Variance in Error between PSO and RPSO

Dim	5		10		20		40	
Algorithm	PSO	RPSO	PSO	RPSO	PSO	RPSO	PSO	RPSO
Griewank	0.00054	1.4E-05	0.00208	9.6E-05	0.00089	3.5E-05	0.00086	4E-05
Sphere2d	0	0	0	0	0	0	5.4E-10	9E-162
Rosenbrock	3.2136	0.15451	2.86199	0.15893	5.62532	3.88094	807.446	590.281
Rastrigin	0.61706	0.0099	11.4249	0.19159	213.775	41.6869	20667.4	1208.52
Total Variance		RPSO	1844.88	PSO	21712.4	%Improvement		84.337

Table 3. Variance in Error between PSO and RPSO-D

Dim	5		10		20		40	
Algorithm	PSO	RPSO-D	PSO	RPSO-D	PSO	RPSO-D	PSO	RPSO
Griewank	0.00158	2.84E-05	0.00131	8.8E-05	0.00042	4.9E-05	0.00097	7.7E-05
Sphere2d	0	0	0	0	0	0	7E-149	7E-189
Rosenbrock	2.49721	4.39E-26	2.84296	0.61767	135.945	5.00172	192.585	142.461
Rastrigin	2.10148	0.019599	42.4373	0.62186	452.966	35.9057	24175.5	761.755
Total Variance		RPSO-D	946.382	PSO	25006.9	%Improvement		92.707

4.1 GBest Behaviour in RPSO-D

As mensioned in sect. 3, the probability of a particle being selected as GBest in RPSO is not uniform. Which means only few particles get the chances to be selected as GBest, which results in stable trajectory of particles. This fact can be observed in Table 4. Results in Table 4 are calculated to show minimum, maximum and average variance in the frequency of each particle of RPSO-D being selected as GBest. The population size 20, dimensions 40 and number of diggers were 3. It can be observed that PSO has extremely low variance on all test functions compared to RPSO-D.

Table 4. Variance in the GBest frequency of PSO and RPSO-D

Function	Algorithm	Min	Max	Average
Griewank	RPSO-D	1560.4	43556.6	6185.6
	PSO	54.0	302.7	131.3
Rastrigin	RPSO-D	2715.4	1494718.5	113461.6
	PSO	47.2	302.6	124.9
Rosenbrock	PSO-D	10608.7	555855.3	136438.1
	PSO	643.8	130190.8	24492.3
Sphere	RPSO-D	39057.5	113022.4	72869.2
	PSO	281.1	1720.7	787.4

5 Conclusion

In classical PSO, all particles follow a single trend and often show premature convergence on the GBest particle. This phenomenon is termed as clustering, which not only inhibits the algorithm to find a better solution but also produce highly variable results on a similar problem. The latter is also algorithm instability. To alleviate these problems, a rank based information dissemination topology has been proposed and particle swarm with this topology was named as Rank based Particle Swarm Optimizer. Empirical analysis proved RPSO is clustering resistant and highly stable.

Clustering helps PSO into finding better optimum when it reaches into the basin of a better optimum. RPSO was found to lack this ability. It was proposed to nominate some particles in RPSO to follow the topology of GBest PSO. The modified RPSO was named as RPSO-D. Experimental results shown that RPSO-D was both clustering resistant as well as capable to quickly refine the optima.

References

1. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: IEEE Int. Conf. on Neural Networks, Perth, Australia, vol. IV, pp. 1942–1948 (1995)
2. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm.theory. In: 6th Int. Symp. on Micro Machine and Human Science, Nagoya, Japan, pp. 39–43 (1995)
3. Maurice, C.: Stagnation Analysis in Particle Swarm Optimization or What Happens When Nothing Happens. Technical Report CSM-460, Department of Computer Science, University of Essex (2006) ISSN: 1744-8050
4. van den Bergh, F., Engelbrecht, A.P.: A New Locally Convergent Parti-cle Swarm Optimiser. In: IEEE Conf. on Systems, Man and Cybernetics, Hammamet, Tunisia, vol. 3, pp. 96–101 (2002)
5. Suganthan, P.N.: Particle Swarm Optimizer with Neighborhood Operator. In: IEEE CEC 1999, Washington DC, USA, pp. 1958–1961 (1999)
6. Kennedy, J.: Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance. In: IEEE CEC 1999, Washington DC, USA, pp. 1931–1938 (1999)
7. Kennedy, J.: Stereotyping: Improving Particle Swarm Performance with Cluster Analysis. In: IEEE CEC 2000, San Diego, USA, pp. 1507–1512 (2000)
8. Lvbgerg, M., Rasmussen, T.K., Krink, T.: Hybrid Particle Swarm Optimizer with Breeding and Subpopulations. In: GECCO 2001, San Francisco, USA (2001)
9. Spears, W.M.: Simple Subpopulation Schemes. In: 1994 Evolutionary Programming Conf. pp. 296–307 (1994)
10. Eberhart, R.C., Simpson, P., Dobbins, R.: Computational Intelligence PC Tools, ch. 6, pp. 212–226. Academic Press Profesional, London (1996)

Self-organized Task Partitioning in a Swarm of Robots

Marco Frison[1,2], Nam-Luc Tran[1], Nadir Baiboun[1,3], Arne Brutschy[1], Giovanni Pini[1], Andrea Roli[1,2], Marco Dorigo[1], and Mauro Birattari[1]

[1] IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
[2] Universitá di Bologna, Bologna, Italy
[3] ECAM, Institut Supérieur Industriel, Brussels, Belgium
mfrison85@gmail.com, nadir_ecam@hotmail.com,
{namltran,arne.brutschy,gpini,mdorigo,mbiro}@ulb.ac.be,
andrea.roli@unibo.it

Abstract. In this work, we propose a method for self-organized adaptive task partitioning in a swarm of robots. Task partitioning refers to the decomposition of a task into less complex subtasks, which can then be tackled separately. Task partitioning can be observed in many species of social animals, where it provides several benefits for the group. Self-organized task partitioning in artificial swarm systems is currently not widely studied, although it has clear advantages in large groups. We propose a fully decentralized adaptive method that allows a swarm of robots to autonomously decide whether to partition a task into two sequential subtasks or not. The method is tested on a simulated foraging problem. We study the method's performance in two different environments. In one environment the performance of the system is optimal when the foraging task is partitioned, in the other case when it is not. We show that by employing the method proposed in this paper, a swarm of autonomous robots can reach optimal performance in both environments.

1 Introduction

Many animal species are able to partition complex tasks into simpler subtasks. The act of dividing a task into simpler subtasks that can be tackled by different workers is usually referred to as *task partitioning* [15].

Although task partitioning may have associated costs, for example because of work transfer between subtasks, there are many situations in which partitioning is advantageous. Benefits of task partitioning include, for example, a reduction of interference between individuals, an improved exploitation of the heterogeneity of the individuals, and an improved transport efficiency [9].

Humans widely exploit the advantages of task partitioning in everyday activities. Through centuries, humans have developed complex social rules to achieve cooperation. These include planning, roles and work-flows. Ancient romans realized the importance of partitioning and they codified it in their military principle *divide et impera* (also known as divide and conquer), which became an axiom in many political [17] and sociological theories [10].

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 287–298, 2010.

Examples of task partitioning can also be observed in social insects. A widely studied case is the foraging task in ants and bees. In foraging, a group of individuals has to collect and transport material to their nest. Foraging involves many different phases and partitioning can occur simultaneously in many of them. Typical phases where task partitioning can occur are the exploration of the environment and the preparation of raw materials [15]. Examples of task partitioning are the harvesting of leaves by the leaf-cutter ants [9], the excavation of nest chambers in *Pogomomyrmex*, and the fungus garden removal in *Atta* [18].

Also in swarm robotics there are situations in which it is convenient to partition a task into subtasks. Advantages include increased performance at group level, stimulated specialization, and parallel task execution. In most of the cases, task partitioning is done a priori and the focus is on the problem of allocating individuals to subtasks in a way that maximizes efficiency. However, in many cases, task partitioning cannot be done a priori because the relevant information on the environment is not available. We consider self-organized task partitioning as a suitable approach in these cases.

In this work, we focus on the case in which a task is partitioned into subtasks that are sequentially interdependent. We propose a simple method, based on individuals' perception and decisions, that allows a swarm of autonomous robots to decide whether to partition a foraging task into subtasks. We test the method with a swarm of simulated robots in two different environmental conditions.

The rest of the paper is organized as follows. In Section 2 we describe the problem and we review related works. In Section 3 we propose an adaptive method that we tested with simulated robots. In Section 4 we provide a description of the experimental framework we consider. In Section 5 we report and discuss the results. Finally, in Section 6 we summarize the contribution of this work and present some directions for future research.

2 Problem Description and Related Works

We study a swarm of robots that has to autonomously decide whether to partition a task into subtasks. Our focus is on situations in which a task is partitioned into sequential subtasks: the subtasks have to be executed in a certain sequence, in order to complete the global task once [5].

In these cases, we can identify *tasks interfaces* where one task ends and another begins. Through tasks interfaces, individuals working on one of the subtasks can interact, either directly or indirectly, with individuals working on other subtasks.

An example of sequential task partitioning, observable in nature, is the foraging activity in *Atta* leaf cutting ants. The sequential interdependency between tasks stems from the fact that each leaf has to be cut from a tree before it can be transported to the nest. Each individual can choose whether to perform both the cutting and transporting subtasks, or to specialize in one subtask only.

Hart at al. [9] described the strategy employed by *Atta* ants: some individuals work on the tree, cutting and dropping leaves to the ground, while the rest of

the swarm gathers and transports these leaves to the nest. Here the advantage of partitioning comes from the fact that the energy cost to climb the tree has to be paid only once by those individuals working as leaf cutters. Disadvantages come from the fact that energy has to be spent to search for leaves on the ground. The task interface can be identified, in this case, as the area where the leaves land. Such areas are usually referred to as *caches*, and facilitate *indirect* transfer of material between individuals. Anderson and Ratnieks described how foragers of different ant species partition the leaf transport task by using caches [4].

Partitioning along foraging trails using *direct* transfer of material between individuals can be observed in other ant species and in other social insects. In the case of direct transfer, the benefit of partitioning can come from the fact that material weight can be matched with the strength of the transporter [2]. Akre et al. observed task partitioning within *Vespula* nectar foraging [1]. Anderson and Ratnieks studied partitioned foraging in honeybees species, showing that the larger the swarm, the higher the performance [3].

Robotic swarms often face situations similar to those of their natural counterparts. However, despite its importance, few works have been devoted to task partitioning in swarm robotics. Notable exceptions are the works of Fontan and Matarić as well as Shell and Matarić on *bucket-brigading* [8,16]. In these works, robots have to gather objects in an environment. Each robot operates in a limited area and drops the object it carries when it reaches the boundaries of its working area. This process leads to objects being passed across working areas until they reach the nest. Lein and Vaughan proposed an extension to this work, in which the size of the robots' working areas is adapted dynamically [11]. Pini et al. showed that the loss of performance due to interference, can be reduced by partitioning the global task into subtasks [14]. To the best of our knowledge, self-organized task partitioning in terms of adaptive task decomposition has never been investigated.

3 The Method

The method we propose allows a swarm of robots to adaptively decide whether to partition a task into sequential subtasks or not. A decision is made by each individual: In case a robot decides to employ task partitioning, it works only on one of the subtasks. In case a robot decides not to employ task partitioning, it performs the whole sequence of subtasks. The method is fully distributed and does not require explicit communication between individuals. The swarm organizes in a way that maximizes the overall efficiency of the group, regardless of the specific environment. Efficiency is defined as the throughput of the system.

In the method proposed, each individual infers whether task partitioning is advantageous or not on the basis of its waiting time at tasks interfaces. We define the probability p that a robot has to employ task partitioning as:

$$p = 1 - \frac{1}{1 + e^{-\theta(w(k))}} \ , \tag{1}$$

with θ being:

$$\theta\big(w(k)\big) = \frac{w(k)}{s} - d \ , \tag{2}$$

and $w(k)$ being the weighted average waiting time at task interfaces after k visits, which is calculated as follows:

$$w(k) = (1 - \alpha)w(k-1) + \alpha w_M \ . \tag{3}$$

In Equation 2, s and d are a scale and a delay factor, respectively. In Equation 3, $\alpha \in (0, 1]$, is a weight factor that influences the responsiveness to changes: higher values lead to a readily responsive behavior. The value of these parameters can be determined empirically. The variable w_M is the *measured waiting time* at task interface and ranges in $[0, w_{MAX})$. The upper limit w_{MAX} ensures that robots eventually renounce to employ task partitioning when their waiting time becomes too high. Each time a robot completes a subtask, it decides whether to employ task partitioning for the next task execution, or not.

4 Experimental Setup

The purpose of the experiments described in this section is to show the validity of the method described in Section 3. To illustrate the approach we have chosen a foraging problem as a testbed. It is a canonical choice in collective robotics as it is easy to model and its applications are numerous [7].

In the experiments, the global task can be described as harvesting objects from a single source and storing them in the nest. The global task can be partitioned into two subtasks, referred to as *harvesting* and *storing*, respectively. Partitioning enables the subdivision of the environment into two areas, linked by a task interface as defined in Section 2. The task interface is represented by a cache that can be used by the robots to exchange objects. As the cache has a limited capacity, robots that decide to use it may have to wait. The waiting time is defined as the delay between the moment when a robot decides to use the cache, either for picking up or dropping objects, and the moment when this effectively becomes possible. It is also possible to avoid the cache by using a separate corridor, which links directly the source and the nest.

Each robot has to autonomously decide whether to partition the foraging task, by using the cache; or not to partition it, by using the corridor. The swarm can also employ a mixed strategy in which some individuals use the cache and others use the corridor. Robots have no notion of the global performance of the swarm. In no case explicit communication is used. Figure 1 shows a simplified state diagram that represents the behavior of each individual.

4.1 Simulation Tools

All the results presented in the paper are obtained using the ARGoS simulation framework [13]. ARGoS is a discrete-time, physics-based simulation environment that allows one to simulate experiments at different levels of detail. For the

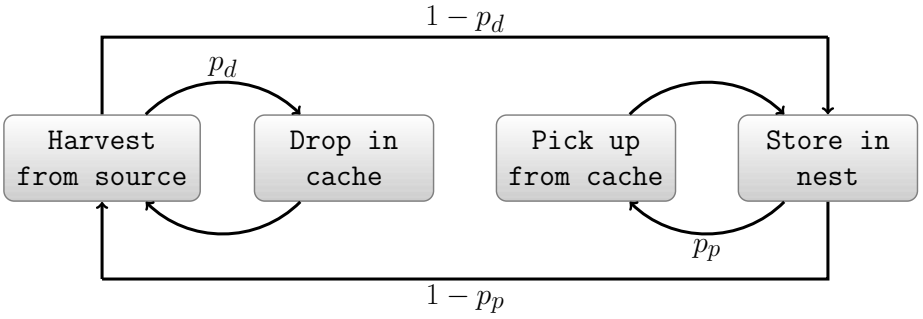


Fig. 1. Simplified state machine representing the behavior of each individual. Probabilities p_d and p_p are both defined using Equation 1 as described in Section 3. The variable p_d represents the probability of using the cache to drop an object. The variable p_p represents the probability of picking up an object from the cache. The states Avoid obstacles and Navigate have been omitted for clarity.

experiments presented in this paper, it is sufficient to simulate kinematics in a bi-dimensional space. A common control interface provides transparent access to real and simulated hardware, allowing the same controller to run also on the real robots without modifications.

The robots we use in this research are the e-pucks[1]. The e-puck has been designed with the purpose of being both a research and an educational tool for universities [12]. ARGoS simulates the whole set of sensors and actuators of the e-puck. In our experiments we use the wheel actuators, the 8 IR sensors for light and proximity detection, the VGA camera, and the ground sensors.

4.2 Harvesting Abstraction

As the e-pucks are not capable of grasping objects, we developed a device to simulate this process [6]. Figure 2 shows a schematic representation of the device, called an *array*[2]. It consists of a variable number of slots, named *booths*. Each booth is equipped with two RGB LEDs that can be detected by the robots through their color camera. A light barrier can detect the presence of a robot within each booth. Reactions to this event, such as changes in LEDs color, are programmable.

In the experiments presented in the paper, a green booth, either in the source or in the cache, means that an object is available there. Analogously, a blue booth means that an object can be dropped there. By using this abstraction, when a robot enters a booth in which the LEDs are lit up in green, we consider that the robot picks up an object from that booth. When a robot enters a booth in which the LEDs are lit up in blue, we consider that the robot drops an object

[1] http://www.e-puck.org/
[2] The array is under development and a working prototype is currently available.

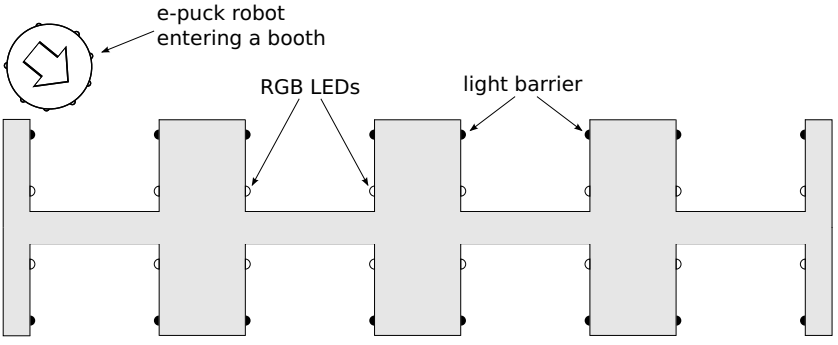


Fig. 2. Schematics of an array of booths with four booths on each side. The light barriers, represented by black semicircles, detect when a robot enters in the corresponding booth. LEDs, used to signal available pick up or drop sites, are represented by blank semicircles.

in that booth. In both cases, when the booth perceives the presence of the robot, it reacts by turning the LEDs red, until the robot has left. Once the robot has left, the booths behave in a different way, depending whether they are source, nest or cache booths. In the case of the source, the booth turns green, to signal the availability of a new object to harvest. In the case of the nest, the booth turns blue, to signal that the corresponding store spot is available again. In case of the cache, if the robot leaves after picking up an object, the booth, previously green, turns off and the corresponding booth on the other side turns blue signaling that the spot is now available again for dropping an object. Conversely, if the robot leaves after dropping an object, the booth, previously blue, turns off and the corresponding booth on the other side turns green signaling that an object is available for being picked up. This simple logic allows us to simulate object transfer through the cache, as well as harvest from the source and store in the nest.

4.3 Environments

We run the experiments in two different environments, named *short-corridor* environment and *long-corridor* environment (see Figures 3a and 3b).

In both the environments, the nest array is located on the right-hand side, while the source array is located on the left-hand side of a rectangular arena. Both the nest and the source arrays have four booths, all on one side. The cache array is located between the nest and the source and has three booths on each side. Therefore, the cache has a limited capacity, which is determined by the number of booths on each side. Different ground colors allow the robots to recognize on which side of the cache they are.

Although the cache array cannot be crossed by robots, a corridor links the two areas and allows the robots to harvest/store objects without using the cache

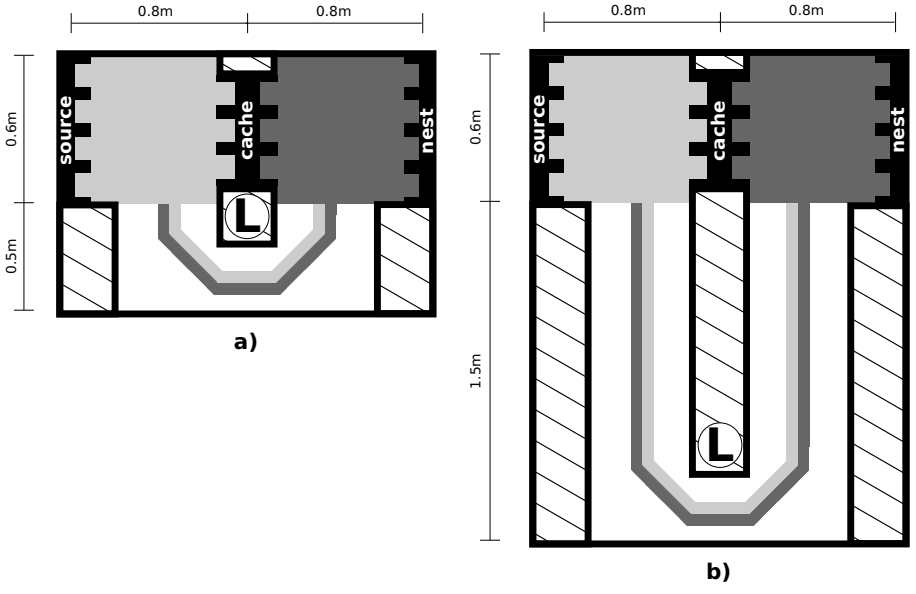


Fig. 3. Representation of the **a)** *short-corridor* and the **b)** *long-corridor* environments used in the experiments. Nest and source arrays have both four booths on one side. The cache array has three booths for each side. Different ground colors help the robots to distinguish between different parts of the environment and to navigate through the corridor that connects the two areas. The light source, used as landmark for navigating in the corridor, is marked with "L".

array (i.e., without partitioning the foraging task). A light source and two colored trails help the robots to navigate through the corridor. Both the environments are 1.6 m wide, but they differ in the corridor's length: in the short-corridor environment the corridor is 1.5 m long, while in the long-corridor environment it is 3.5 m long. Both the use of the cache and the use of the corridor entail costs. The cache can be seen as a shortcut between source and nest: robots cannot cross the cache, but its use can make material transfer faster. However, the cache can also become a bottleneck as the decision of using it can lead to delays if it is busy when dropping objects, or empty when picking them up. The decision of using the corridor imposes a cost due to the time spent traveling through it. Thus, the transfer cost varies with cache and group size while the travel cost varies with corridor length. As we keep the size of the cache array constant in our experiments, the corridor length determines the relative cost between partitioning and not partitioning. In the long-corridor environment, the use of the cache is expected to be preferable. On the other hand, in the short-corridor environment, the cost imposed by the corridor length is low and should lead to the decision not to use the cache.

4.4 Experimental Settings

We run two different sets of experiments: in a first set of experiments we are interested in assessing the performance of the adaptive method that we propose, while in the second set of experiments we aim at evaluating its scalability. In the first set of experiments the robotic swarm is composed of twelve e-pucks, each controlled by the same finite state machine depicted in Figure 1. In both the environments, we compare the adaptive method described in Section 3 to two strategies which always partition ($p_d = p_p = 1$) or never partition ($p_d = p_p = 0$). These are used as reference strategies for evaluating both the performance of the proposed method and its capability of adapting to different environmental conditions. The values of the parameters have been determined empirically and fixed to $s = 20$, $d = 5$, $\alpha = 0.75$, $w_{MAX} = 15\,\text{s}$. The duration of each experimental run is 150 simulated minutes. For each experimental condition we run 30 repetitions, varying the seed of the random numbers generator. At the beginning of each experiment the robots are randomly positioned in the right side of the environment, where the nest array is located. As the average waiting time $w(k)$ is initially equal to zero, all the robots start with probabilities $p_d = p_p \approx 1$: from Equations 1 and 2, with $d = 5$; when $\theta\big(w(k)\big) = 0$, $p = 0.993$.

In the second set of experiments we compare the adaptive method to the reference strategies in the short-corridor environment. In this case the size of the swarm varies in the interval $[4, 60]$. For each condition we run 10 randomly seeded experiments. The remaining parameters of the experiment are the same as described for the first set of experiments.

5 Results and Discussion

As we keep the capacity of the cache array constant, it is the length of the corridor that determines which behavior maximizes the throughput of the system. Partitioning allows the robots to avoid the corridor but, in order to exploit efficiently the cache, the swarm has to organize and to work on both of its sides. Additionally, as pointed out in Section 4, the robots might have to wait in order to use the cache.

In the short-corridor environment, the cost of using the cache is higher than the cost of using the corridor. In this case the robots should decide for a non-partitioning strategy, without exchanging objects at the cache. Conversely, in the long-corridor environment the time required to travel along the corridor is high, and partitioning the task by using the cache is expected to be more efficient.

The graphs in Figure 4 show the average throughput of a swarm of twelve robots in the two environments. Throughput is measured as the number of objects retrieved per minute. The adaptive method is compared with the two reference strategies in which the robots never/always use the cache. As expected, each of these reference strategies performs well only in one environment: the strategy that never uses the cache performs better in the short-corridor environment, while the strategy that always use the cache performs better in the

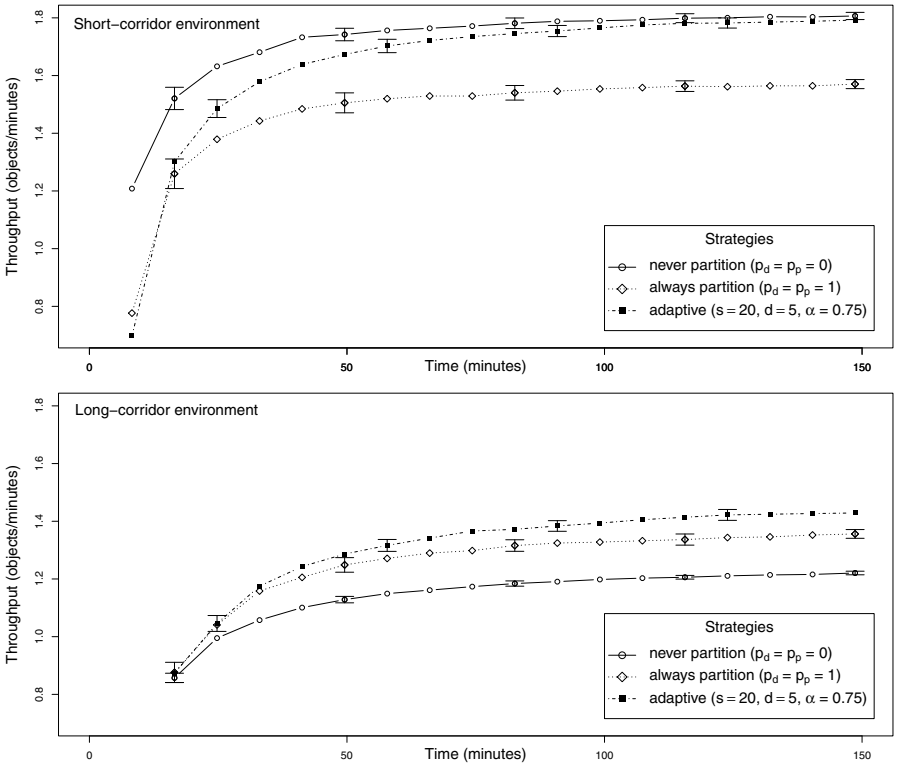


Fig. 4. Average throughput in the short-corridor (top) and long-corridor (bottom) environment for different strategies. Time is given in simulated minutes. Throughput is measured as objects retrieved per minute. Parameters for the adaptive behavior are set to $s = 20$, $d = 5$, $\alpha = 0.75$, $w_{MAX} = 15\,\mathrm{s}$. Parameter values have been obtained empirically. Each experiment has been repeated 30 times, varying the seed of the random number generator. The bars around the single data points represent the confidence interval of 95% on the observed mean.

long-corridor environment. On the other hand, the adaptive method we propose shows good performance in both environments.

Concerning the long-corridor environment, we assumed that the best strategy was to always use the cache and to avoid the corridor. However, Figure 4(bottom) shows that the adaptive method proposed improves over this fixed strategy. To better understand this behavior, we empirically determined the fixed-probability strategy yielding the highest throughput for each environment. This analysis revealed that the best strategy in the long-corridor environment is to use the cache with a probability around 80%. In the short corridor environment a non-partitioning strategy is preferable.

Table 1 reports the average throughput obtained at the end of the run for each strategy in each environment. The results reported in the table show that

Table 1. Average throughput at the end of the experiment for a swarm of 12 robots. Results are reported for different partitioning strategies in the short-corridor and long-corridor environments. The values in parenthesis indicate the 95% confidence interval on the value of the mean.

	Never partition	Always partition	Fixed ($p_d, p_p = 0.8$)	Adaptive
Short-corridor	1.81 (±0.012)	1.57 (±0.015)	1.67 (±0.016)	1.79 (±0.017)
Long-corridor	1.22 (±0.006)	1.36 (±0.015)	1.40 (±0.013)	1.43 (±0.017)

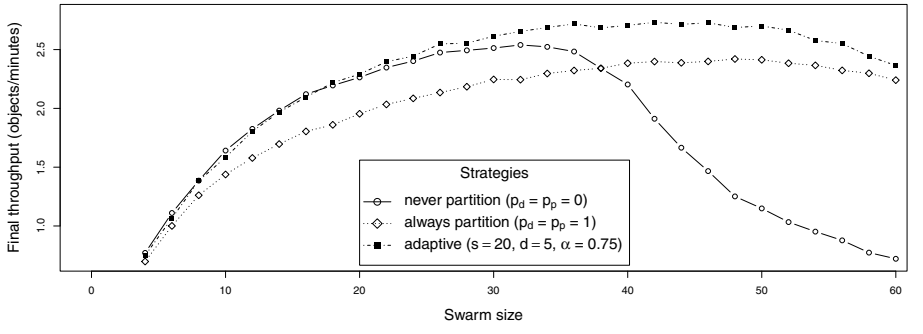


Fig. 5. Impact of the swarm size in the short-corridor environment. The value of the throughput, measured as number of objects retrieved per minute, is the average value reached by the swarm at the end of the experiment. Each experimental run lasts 150 simulated minutes. For each experimental condition we run 10 repetitions, varying the seed of the random number generator.

fixed-probability strategies perform well only in one of the two environments. Our adaptive method, on the other hand, reaches good performance in both the short-corridor and the long-corridor environment. These results confirm that the method we propose allows a swarm of robots to take a decision concerning whether to partition a task into sequential subtasks or not.

The graph in Figure 5 shows the results of the second set of experiments, in which we focus on scalability. In this case we compare different strategies for different swarm sizes in the short-corridor environment. As discussed previously, in the short-corridor environment the optimal strategy is to always use the corridor. It can be observed that for small swarm sizes this strategy performs well. However, the performance of this strategy drops drastically when the number of robots increases. The reason for this degradation is the increasing interference between the robots, which increases the cost of using the corridor. The partitioned strategy does not suffer from steep drops of performance. However, the throughput is considerably lower for smaller group sizes, as the waiting time at the cache becomes dominant. The adaptive method performs well across all the studied swarm sizes, finding a good balance between the robots that use the cache and those that use the corridor.

6 Conclusions

In this research we investigated the self-organized task partitioning problem in a swarm of robots. In particular we have proposed an adaptive method to tackle the task partitioning problem with a simple strategy based on individual's perception of each subtask performance. In the method proposed, the subtask performance is estimated by each robot by measuring its waiting time at task interfaces. Results show that the adaptive method we propose reaches the best performance in the two environments we considered, employing task partitioning only in those cases in which the benefits of partitioning overcome its costs. The study of the impact of the group size reveals that the method scales well with the swarm size. Future work will concern the study of self-organized task partitioning in multi-foraging problems in environments with several caches. In addition, we are interested in studying the application of the method proposed to cases in which partitioning happens through direct material transfer.

Acknowledgements. This work was partially supported by the virtual Swarmanoid project funded by the Fund for Scientific Research F.R.S.–FNRS of Belgium's French Community. Marco Dorigo and Mauro Birattari acknowledge support from the F.R.S.–FNRS, of which they are a research director and a research associate, respectively. Marco Frison acknowledges support from "Seconda Facolt di Ingegneria", Alma Mater Studiorum, Università di Bologna. Andrea Roli acknowledges support from the "Brains (Back) to Brussels" 2009 programme, founded by IRSIB – Institut d'encouragement de la Recherche Scientifique et de l'Innovation de Bruxelles.

References

1. Akre, R.D., Garnett, W.B., MacDonald, J.F., Greene, A., Landolt, P.: Behavior and colony development of *Vespula pensylvanica* and *Vespula atropilosa* (hymenoptera: Vespidae). Journal of the Kansas Entomological Society 49, 63–84 (1976)
2. Anderson, C., Jadin, J.L.V.: The adaptive benefit of leaf transfer in *Atta colombica*. Insectes Sociaux 48, 404–405 (2001)
3. Anderson, C., Ratnieks, F.L.W.: Task Partitioning in Insect Societies. I. Effect of colony size on queueing delay and colony ergonomic efficiency. The American naturalist 154(5), 521–535 (1999)
4. Anderson, C., Ratnieks, F.L.W.: Task partitioning in insect societies: novel situations. Insectes Sociaux 47, 198–199 (2000)
5. Brutschy, A.: Task allocation in swarm robotics. Towards a method for self-organized allocation to complex tasks. Tech. Rep. TR/IRIDIA/2009-007, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2009)
6. Brutschy, A., Pini, G., Baiboun, N., Decugnière, A., Birattari, M.: The IRIDIA TAM: A device for task abstraction for the e-puck robot. Tech. Rep. TR/IRIDIA/2010-015, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2010)
7. Dorigo, M., Sahin, E.: Guest editorial. Special issue: Swarm robotics. Autonomous Robots 17(2-3), 111–113 (2004)

8. Fontan, M.S., Matarić, M.J.: A study of territoriality: The role of critical mass in adaptive task division. In: From Animals to Animats 4: Proceedings of the Fourth International Conference of Simulation of Adaptive Behavior, pp. 553–561. MIT Press, Cambridge (1996)
9. Hart, A.G., Anderson, C., Ratnieks, F.L.W.: Task partitioning in leafcutting ants. Acta Ethologica 5, 1–11 (2002)
10. Kant, E.: Perpetual Peace: A Philosophical Sketch. Hackett Publishing Company, Indianapolis (1795)
11. Lein, A., Vaughan, R.: Adaptive multi-robot bucket brigade foraging. In: Bullock, S., Noble, J., Watson, R., Bedau, M.A. (eds.) Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems, pp. 337–342. MIT Press, Cambridge (2008)
12. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, Portugal, pp. 59–65, IPCB: Instituto Politècnico de Castelo Branco (2009)
13. Pinciroli, C.: Object retrieval by a swarm of ground based robots driven by aerial robots. Tech. Rep. TR/IRIDIA/2007-025, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2007)
14. Pini, G., Brutschy, A., Birattari, M., Dorigo, M.: Interference reduction through task partitioning in a robotic swarm. In: Filipe, J., Andrade-Cetto, J., Ferrier, J.L. (eds.) Sixth International Conference on Informatics in Control, Automation and Robotics – ICINCO 2009, pp. 52–59. INSTICC Press, Setbal (2009)
15. Ratnieks, F.L.W., Anderson, C.: Task partitioning in insect societies. Insectes Sociaux 46(2), 95–108 (1999)
16. Shell, D.J., Matarić, M.J.: On foraging strategies for large-scale multi-robot systems. In: Proceedings of the 19th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2717–2723 (2006)
17. Traiano, B.: La Bilancia Politica di Tutte le Opere di Traiano Boccalini, Part 2-3. Kessinger Publishing, Whitefish, Montana, USA (1678)
18. Wagner, D., Brown, M.J.F., Broun, P., Cuevas, W., Moses, L.E., Chao, D.L., Gordon, D.M.: Task-related differences in the cuticular hydrocarbon composition of harvester ants. *Pogonomyrmex barbatus*. J. Chem. Ecol. 24, 2021–2037 (1998)

Slime Mold Inspired Path Formation Protocol for Wireless Sensor Networks*

Ke Li[1], Kyle Thomas[2], Claudio Torres[3], Louis Rossi[3], and Chien-Chung Shen[1]

[1] Computer & Info. Sciences
[2] Chemical Engineering
[3] Mathematical Sciences
University of Delaware, Newark, DE, USA
{kli,cshen}@cis.udel.edu, kcthomas@udel.edu, {torres,rossi}@math.udel.edu

Abstract. Many biological systems are composed of unreliable components which self-organize efficiently into systems that can tackle complex problems. One such example is the true slime mold *Physarum polycephalum* which is an amoeba-like organism that seeks food sources and efficiently distributes nutrients throughout its cell body. The distribution of nutrients is accomplished by a self-assembled resource distribution network of small tubes with varying diameter which can evolve with changing environmental conditions without any global control. In this paper, we use a phenomenological model for the tube evolution in slime mold and map it to a path formation protocol for wireless sensor networks. By selecting certain evolution parameters in the protocol, the network may evolve toward single paths connecting data sources to a data sink. In other parameter regimes, the protocol may evolve toward multiple redundant paths. We present detailed analysis of a small model network. A thorough understanding of the simple network leads to design insights into appropriate parameter selection. We also validate the design via simulation of large-scale realistic wireless sensor networks using the QualNet network simulator.

1 Introduction

One fundamental design issue in wireless sensor networks (WSNs) is, given an arbitrary deployment of sensor nodes, how to connect the source nodes to the sinks so that data collected by the source nodes flow efficiently and reliably to the sinks. To facilitate such data transfer, a path formation protocol becomes essential that will determine multi-hop routes between source nodes and sink nodes. We avoid strategies that rely upon global coordination and optimization because such approaches will not scale well in large networks or networks where nodes are added or deleted dynamically, or where connections form or break easily. To seek sound solutions that rely upon local interactions, we borrow mechanisms from similar problems in the natural world and adapt them to suit the challenges of sensor networks. The natural world is full of resource distribution networks

* Preliminary work of the paper appeared as a 2-page poster in the 3rd IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems (SASO 2009).

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 299–311, 2010.

such as circulatory, respiratory and nervous systems in animals that are elegant solutions to very specific and complex problems. We draw our inspiration from true slime mold *Physarum polycephalum*, a biological system that has properties more aligned with the requirements of sensor networks. In particular, the dynamics of *Physarum polycephalum* constructs resource distribution networks in response to environmental conditions. This is distinct from circulatory systems which are hard-wired for a particular physiology.

Slime mold *Physarum polycephalum* is a multinuclear, single-celled organism that has properties making it ideal for the study of resource distribution networks [1]. While the organism is a single cell, slime mold can grow to tens of centimeters in size so that it can be studied and manipulated with modest laboratory facilities. The presence of nutrients in the cell body triggers a sequence of chemical reactions leading to oscillations along the cell body [5]. Tubes self-assemble which are perpendicular to the oscillatory waves to create networks linking nutrient sources throughout the cell body. There are two key mechanisms in the slime mold life cycle that transfer readily to WSNs. First, during the growth cycle, the slime mold will explore its immediate surroundings with pseudopodia via chemotaxis to discover new food sources. We have applied this mechanism successfully to WSNs in a previous work treating data sources and sinks as singular potentials [2]. In this current work, we exploit the second key mechanism which is the temporal evolution of existing routes through nonlinear feedback to efficiently distribute nutrients throughout the organism. In slime mold, it can be shown experimentally that the diameters of tubes carrying large fluxes of nutrients grow to expand their capacity, and tubes that are not used decline and can disappear entirely.In the context of WSNs, the diameter of a tube may be used to denote the Quality of Service (QoS) of a wireless channel.

To evaluate the merits of path formation, we focus on three metrics that directly measure the network performance: expected hop count, fault tolerance, and node degree distribution. The *expected hop count* denotes the efficiency of data flow in a network by measuring the expected number of nodes a data packet must traverse to reach the sink. We compute the expectation because the protocol may construct topologies with multiple disjoint routes. The *fault tolerance* is a gross indicator of network robustness measured by the probability that a node failure will disconnect at least one source from the network. Finally, we measure the *node degree distribution* for the network as an indicator of connectivity.

This paper is structured as follows. In Section 2, we describe the model of Tero et al. for solutions of *Physarum polycephalum* to mazes and map it to the path formation problems of WSNs. In Section 3, we describe the path formation protocol based on the slime mold model. In Section 4, we verify our model by independently computing all possible stationary states and their stability on a very small network. In Section 5, we perform large scale simulations and measure the protocol performance with quantitative metrics of expected hop count, fault tolerance, and node degree distribution as functions of network parameters.

2 A Phenomenological Model for Slime Mold

We base our path formation protocol for WSNs on the phenomenological model proposed by Tero et al. for tube evolution in *Physarum polycephalum* to reproduce the slime mold maze solving experiments [3,6]. In recent article on applications of their slime-inspired algorithm, Tero et al. independently acknowledge the suitability of this type of model for sensor networks [7]. The model captures the evolution of tube capacities in an existing network through a dynamical system as follows. The original model consists of a set of nodes representing either a food source or a junction in the maze connected by a set of edges. Along each edge, the flow of nutrients from node i to node j is represented as q_{ij}, which is directional and $q_{ij} = -q_{ji}$. Flow through the network is driven by a pressure at each node p_i. The cross-sectional size of each tube is represented by D_{ij} ($D_{ij} = D_{ji}$). At each node, a Kirchhoff law is required to enforce conservation of nutrient flux at each junction. All above variables evolve in time as follows:

$$q_{ij} = \frac{D_{ij}}{L_{ij}}(p_i - p_j), \tag{1a}$$

$$\sum_{j \in N_i} q_{ij} = m_i, \tag{1b}$$

$$\frac{dD_{ij}}{dt} = f(|q_{ij}|) - rD_{ij}, \tag{1c}$$

where L_{ij} is the length of the tube, N_i represents the set of neighbors for node i, m_i is the amount of nutrient flux into the network from i, $f(x)$ is either an exponential or sigmoidal function describing the rate of growth of the tube size in response to nutrient flow rate, and r is a tube size linear decay rate. The system (1) contains a continuity equation (Kirchhoff Law) (1b) balancing the flow out of a node with the source value. In Tero et al. [6], nutrient sources were designated to be $m_i = 1$ at the entrance or $m_i = -1$ at the exit to drive flow through a maze. At junctions within the maze, $m_i = 0$.

We use the Tero model as a foundation for the path formation protocol for WSNs. While the distinction of identical nutrient sources as either $m_i = +1$ or $m_i = -1$ in the slime mold seems arbitrary, this aspect of algorithm maps well to WSNs which have sensors that are data sources and sinks that represent highly capable nodes collecting source data. The set N_i of neighbors denotes the set of nodes within the transmission radius of node i. Thus, there is a communication link between node i and all members of N_i. In the proposed protocol, we assume symmetric links with all identical transmission radii. While the tube length L_{ij} could be a useful tuning parameter mappable to special restrictions on network connections, we do not pursue these issues in this paper and simply assume equal L_{ij}'s throughout the network. We represent $f(x)$ as sigmoid function slightly differently from Tero et al. as

$$f(x) = rD_{\max}\frac{a|x|^{\mu}}{1 + a|x|^{\mu}}, \tag{2}$$

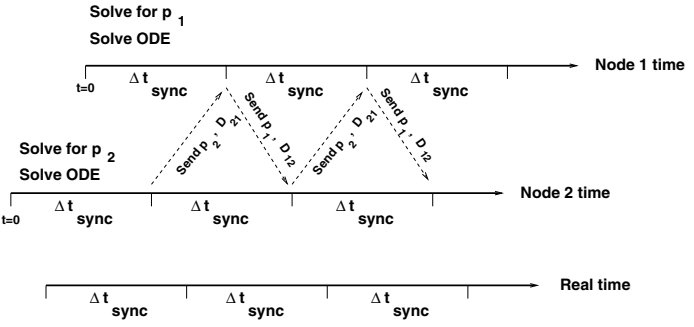


Fig. 1. Schematic diagram of local computation followed by synchronization of protocol variables. Nodes 1 and 2 perform computations independently within a fixed cycle of Δt_{sync}, and share information afterward.

where $D_{\max}$ specifies the upper limit on D_{ij}'s. The exponent μ plays a critical role in the behavior of this model, which will be discussed later. The parameter a describes the shape of the sigmoid and has little impact on network performance. Without loss of generality, we simply select all L_{ij}'s, r, D_{max}, and a to be 1 when calculating numerical solutions to the model as well as simulating the path formation protocol.

Like ant colony optimization (ACO), the slime mold model includes dynamic positive and negative feedback mechanisms (1c). The key distinction is that the model permits nearly simultaneous coordination through flow continuity (1a,1b), which provides both the advantage that more information is available, and the disadvantage that the pressure equation must be solved iteratively as will be discussed in Section 3.2. The solution to this system may be used to move data from sources to data sink(s) in WSNs following a multi-path routing scheme. The outward fluxes from a node i guide how packets are routed through the network. If we denote O_i to be the subset of $k \in N_i$ such that $q_{ik} > 0$, O_i then represents the next-hop neighbors of node i on a route toward the sink. We define the ratio

$$w_{ij} = \frac{q_{ij}}{\sum_{k \in O_i} q_{ik}}. \tag{3}$$

If node i has C data packets to send, Cw_{ij} packets (rounding as necessary) would be sent to node j for each $j \in O_i$. While the w_{ij}'s do not necessarily represent probabilities, we could pose a stochastic routing protocol by routing each packet arriving at node i to node j with probability w_{ij}.

3 Protocol Description

Although model (1) is globally coupled via the pressure field, the inspired path formation protocol solves the model in a distributed manner. Each node i maintains its local states of own pressure p_i, perceived neighbor pressures p_j's, and

tube size D_{ij}'s for all $j \in N_i$. We initialize pressures uniformly as 1.0 (except for the sink as 0), with D_{ij}'s randomly selected between 0 and $D_{\max}$. Each node then performs local computations in a fixed cycle of Δt_{sync}, solving model (1b,1c) for pressure and link diameters. At the end of each Δt_{sync} cycle, a node broadcasts its latest computing results to neighbors, which allows globally coupled information to propagate across the network. Meanwhile, each node keeps updating perceived neighbor pressures to be used in future local computation. The loop of local computation followed by states synchronization at fixed interval of Δt_{sync} is demonstrated in Figure 1. Notice that the localized path formation protocol solves model (1) cooperatively among neighboring nodes and in real time without explicit clock synchronization across the network. All that is required is that clocks run forward at approximately the same rate in every node.

3.1 Local Data Structures

Each node i locally maintains a node type ($nodeType_i$), a net flux ($netFlux_i$), a pressure value (p_i), and a *neighbor* table ($nTab_i$). The value of $nodeType_i$ could be *SINK*, *SOURCE*, or *OTHER* (as potential relays). The $netFlux_i$ specifies the pure flux m_i flowing out of node i into the network, which is positive at any data source, negative at the data sink, and zero at any other node. The pressure value of each node other than the sink starts with 1.0 and evolves based on model (1), while the sink's pressure remains 0 at all time. The *neighbor* table entry $nTab_i(j)$ corresponds to the 1-hop neighbor j of node i, with the form of $\langle p_j, D_{ij}, q_{ij}, L_{ij} \rangle$. The *neighbor* table keeps track of local dynamics of the evolving system and thus forms the basis for solving model (1).

3.2 Local Computation

Between synchronizations of pressures and link diameters with neighbors, each node i performs local computation in R rounds to solve model (1) for pressure p_i, neighbor link flux q_{ij}'s, and diameter D_{ij}'s, based on the following mathematical methods.

To solve for p_i locally, we substitute the flow model (1a) into the pressure equation (1b):

$$p_i \leftarrow \frac{m_i + \sum_{j \in N_i} \frac{D_{ij}}{L_{ij}} p_j}{\sum_{j \in N_i} \frac{D_{ij}}{L_{ij}}}. \tag{4}$$

Similar to the Jacobi algorithm for solving diagonally dominant linear systems, the above method of solving Kirchhoff law (1b) in terms of pressure is *translation invariant*, meaning that one can take a pressure solution and add the same constant to all pressures to obtain another solution. Therefore, we fix the pressure to be 0 at the sink node which does not satisfy Kirchhoff's law. Conservation of flux is already imposed and fixing the pressure at the sink node guarantees a unique pressure solution. With (1b) satisfied at all nodes except for the sink, we

then numerically solve the evolution equation for D_{ij} iteratively using the first order implicit scheme proposed by Tero:

$$D_{ij}^{n+1} = \frac{D_{ij}^{n} + \Delta t f(|q_{ij}^{n}|)}{1 + \Delta t r}, \tag{5}$$

where the fluxes q_{ij} are determined from the solved pressures p_i, Δt is the duration of one discrete time step (or iteration), and D_{ij}^{n} is the value of D_{ij} at the n^{th} iteration.

Each round of the local computation works as follows. First, every node i except for the sink calculates its pressure p_i by solving the linear equation (Kirchhoff's law), given neighbors' p_j's and D_{ij}'s based on (4), while the sink remains zero pressure as the minimum pressure level. Next, for each of its neighbor j, node i determines the corresponding flux q_{ij} with the freshly calculated p_i. Then it solves the adaptive system (1c) for D_{ij} based on (5) in S iterations, with each iteration's result based on the previous one and fed into the next. Since each iteration signifies Δt time, the total time for R rounds of S iterations is thus $T = R \times S \times \Delta t$ before the next synchronization. We require that $\Delta t_{\text{sync}} \geq T$.

3.3 Synchronization

Before each cycle of local computation, every node i shares its pressure p_i and synchronizes D_{ij}'s with neighbors by broadcasting a one-hop synchronization (SYNC) packet. The SYNC packet contains the sender's *id* and current pressure, together with the latest *neighbor-diameter* table, including the total number of entries (*#entries*) followed by each entry $\langle n_j, D_{id,n_j} \rangle$. In particular, D_{id,n_j} represents the diameter of link to its neighbor n_j, retrieved from the sender's *neighbor* table. The initial synchronization also serves as a hello packet for neighbor discovery, whose *neighbor-diameter* table is empty with $\#entries == 0$.

Upon receiving a SYNC packet, each node tries to synchronize its *neighbor* table in terms of neighbor pressure and link diameter based on information from the packet. In case of the first time to hear from sender j, node i will insert in its *neighbor* table a new entry for j. Specifically, D_{ij} in the new entry is either assigned as the corresponding D_{ji} from the packet's *neighbor-diameter* table if it exists, or otherwise initialized as a random value $rand(i + j)$ within $[0, D_{max}]$. Notice that the latter case applies when receiving the initial batch of SYNC packets which also serve for neighbor discovery, and the random diameter uniquely dependent on $(i+j)$ ensures that $D_{ij} = D_{ji}$ at the start. In case sender j already exists in node i's *neighbor* table, p_j is updated and D_{ij} synchronized as the mean of itself and D_{ji} from the packet.

3.4 Issue of Asymmetric Neighborhood

Due to unreliability of wireless communications, synchronization packets may get lost or corrupted. This is annoying when lost packets assist in neighbor discovery, which could cause "asymmetric neighborhood", *e.g.*, node i can hear (sometimes) from node j thus recording j in its *neighbor* table and counting p_j

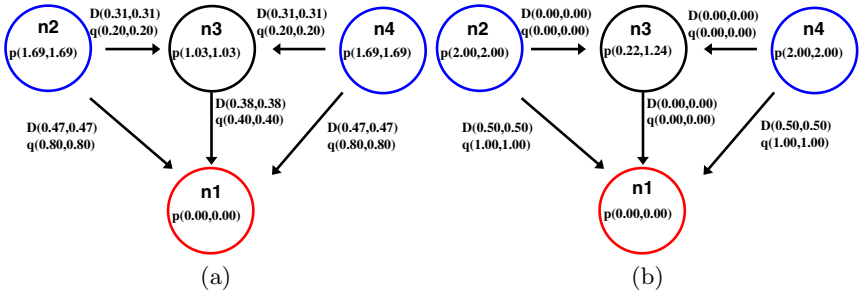


Fig. 2. Comparison between steady states from numerical solutions (left value) and QualNet simulations (right value) on the 4-node network under different μ values. (a) $\mu = 0.5$, (b) $\mu = 2$.

and D_{ij} during its local computations, while node j could not receive successfully from i thus never taking i as its neighbor. This problem is aggravated when node i decides through local computation to send majority of its outgoing flux to its seeming neighbor j, while node j is ignorant of the large flux from node i, which could potentially disrupt the global flow continuity (1b) and render the protocol unable to converge. Asymmetric neighborhood becomes widespread as the number of nodes within transmission range increases which causes severer channel contentions.

Although packet loss may be mitigated via MAC layer schemes like random jitters, it could never be eliminated as long as the channel is unreliable. We take an alternative approach to contain the bad effect of packet loss so as to prevent the asymmetric neighborhood, by introducing the neighborhood threshold (*neighThresh*). Every node tracks the number of SYNC packets sent out (*#pktSent*), as well as received from each neighbor (*#pktRcvdt*) recorded within each neighbor entry of its *neighbor* table. During local computation, nodes only consider "true" neighbors that satisfy:

$$\#pktRcvd > \#pktSent \times neighThresh \tag{6}$$

where $neighThresh \in [0, 1]$, and needs to be carefully selected. In practice, the neighborhood threshold is usually chosen within $[0.5, 1]$ — The more contentions or worse channel quality, the bigger the threshold used. The basic rationale of this approach is the assumption that the channel quality between two nodes is roughly symmetric when both nodes transmit using the same power within a short time period.

4 Validation with Stability Analysis

4.1 Validation

To validate the path formation protocol, we systematically studied the stationary solutions of the dynamical system (1) on a 4-node small network, and compared

exact numerical solutions with results from distributed calculations in realistic QualNet [4] simulations. Details on calculating exact solutions are discussed in Section 4.2.

The 4-node network consists of one sink n_1, two sources n_2 and n_4 each with 1 unit of flux to be sent to the sink, and one potential relay n_3, with network layout shown in Figure 2. By numerically solving the model, we found 9 steady states for $\mu = 0.5$ (demonstrated in Figure 3 inside ovals), and 6 for $\mu = 2$.We ran QualNet simulations under same set of parameters with different random seeds. For $\mu = 0.5$, all trials converged to the same single steady state shown in Figure 2(a), the reason of which will be discussed in Section 4.2. For $\mu = 2$, different steady states were found by simulations. Figure 2 compares one corresponding steady state between the solution of numerical method and QualNet simulation for each μ value. We find that the exact numerical solutions and the realistic QualNet simulation results agree on D_{ij}'s and q_{ij}'s. So do they on almost all pressure p values except for node n_3 when $\mu = 2$, in which state both sources n_2 and n_4 send all fluxes directly to the sink n_1 without passing through n_3. Therefore n_3 has the "freedom" of choosing any positive value as pressure, since it will not be used for relay. This "freedom" doesn't impact the protocol because it only happens when the flux through the node is zero. From the mathematical point of view, this case corresponds to the existence of a nontrivial, homogeneous solution when we solve (1a) and (1b) for the pressure.

4.2 Linear Stability Analysis

To understand the stability of solutions to model (1), we analyze solutions to the nonlinear system of equation $\mathcal{F}(D_{ij}) = f(|q_{ij}|) - rD_{ij} = 0$ when setting $\frac{dD_{ij}}{dt} = 0$ in (1). The stationary solution to $\mathcal{F}(D_{ij}) = 0$ is defined as an implicit nonlinear system, where the implicit part (q_{ij}) is obtained by solving a nested linear system for pressure values using system (1). After calculating solutions to $\mathcal{F}(D_{ij}) = 0$, we then compute the Jacobian matrix $\left[\frac{\partial \mathcal{F}_{ij}}{\partial D_{kl}}\right]$ evaluated at the stationary point of interest. The eigenvalues and eigenvectors of each solution determine the stability of the system. If all eigenvalues are negative, the

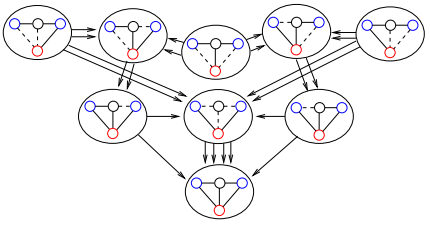


Fig. 3. Stability diagram for the 4-node network with $\mu = 0.5$. The topology of each of these 9 steady solutions is shown inside the oval, with solid lines indicating links with $D_{ij} \neq 0$ and dotted lines for links with $D_{ij} = 0$. Each pair of arrows represents the growth of a small positive and negative amplitude perturbation along an eigenvector.

solution is a stable steady state. Otherwise when there are any positive eigenvalues, the solution is unstable, which will evolve to a different steady state, with the linear evolution described by the eigenvector corresponding to the positive eigenvalue.

The complete understanding of small problems gives us useful insights into how parameter values affect the behavior of the whole network. In particular, the parameter μ dominates the evolution of links in the network. Generally speaking, for $\mu = 2$ (in general $\mu > 1$) all single route solutions with positive D_{ij} values are stable. On the other hand, for $\mu = 0.5$ (in general $\mu < 1$) only the solution where all links are used is stable, which corresponds to the single steady state reached by QualNet simulations. Since there are both stable and unstable numerical solutions when $\mu = 0.5$, we also built the stability diagram over all 9 solutions shown in Figure 3, where the perturbations were computed using the eigenvectors related to the positives eigenvalues. We find that for $\mu = 0.5$ the initialization does not affect the final stable state at all, which should always be the all-link solution illustrated by Figure 2(a). This mathematical analysis also explains why all QualNet simulations converged to this stable solution when $\mu = 0.5$. Finally, these general properties transfer to larger networks.

5 QualNet Simulation

We performed QualNet simulations on 100 and 300-node networks with one sink and varying numbers of sources. All nodes were random distributed over a terrain of size 1000×1000 m^2. Each node in the network was equipped with a radio transceiver capable of transmitting signals up to approximately 80 meters. The underlying wireless channel had a data rate of 2 Mbps, using the two-ray path loss model without fading. IEEE 802.11 DCF was used as the MAC layer protocol, and IP as the network layer on top of which our protocol ran. No application traffic was employed since the sole purpose was to connect source nodes to the data sink. The neighborhood threshold is 0.8 to avoid neighborhood asymmetry. For each cycle of local computation, the number of round $R = 10$, iteration $S = 1$, and $\Delta t_{\text{sync}} = 0.01$ sec. The following three quantitative metrics are designed to measure qualities of resulting network connectivity under $\mu = 0.5$ and $\mu = 2$ with different portions of sources:

- **Expected hop count:** indicates the *efficiency* of the connectivity in terms of hop count distance between a source and the sink. The smaller the hop count, the higher the efficiency. Let $W = [w_{ij}]^T$ be the Markov transition matrix with w_{ij} defined in Equation (3). Let $\boldsymbol{e}_l$ be a unit vector consisting of all zeros except at element l. Then $W\boldsymbol{e}_l$ is a vector whose j^{th} element denotes the probability of a data packet starting from node l and arriving at node j in one hop. Similarly, $W^k\boldsymbol{e}_l$ is a vector of probabilities for k hops. Therefore, the expected hop count from source node l to sink node s is

$$E(l, s) = \sum_{k=1}^{\infty} kW^k\boldsymbol{e}_l. \tag{7}$$

Table 1. Fault tolerance of a 300-node network with 1 sink and varying numbers of sources

source	10%		20%		30%		40%		50%	
μ	0.5	2	0.5	2	0.5	2	0.5	2	0.5	2
1-fault	1	.9851	.9916	.9874	.9856	.9761	.9832	.9609	.9799	.9530
2-fault	.9998	.9700	.9826	.9739	.9709	.9616	.9659	.9222	.9595	.9070

- **Fault tolerance:** indicates the *robustness* of the connectivity. We define the x-fault tolerance to be the probability that removing x relay nodes will *not* yet result in any source being disconnected from the sink. The higher the probability, the better the robustness.
- **Degree distribution:** defines $P(d)$ as the probability of a node in the network having a degree of d. The presence of a small number of high degree nodes may indicate a greater dependence on a few "gateway nodes" for network connectivity.

Figure 4 shows snapshots of steady state connectivity under $\mu = 0.5$ and $\mu = 2$ over a 100-node randomly generated network with one data sink and three data sources. Figure 5 depicts a 300-node network with one data sink and thirty data sources. The width and color of a link signify the value of corresponding flow q_{ij}. The arrow of the link shows the direction of the flow. We find that the network with $\mu < 1$ (e.g., $\mu = 0.5$) evolves towards multi-route robust connectivity, whereas the network with $\mu > 1$ (e.g., $\mu = 2$) evolves into single-route efficient topology. This verifies that the exponent μ is a critical parameter

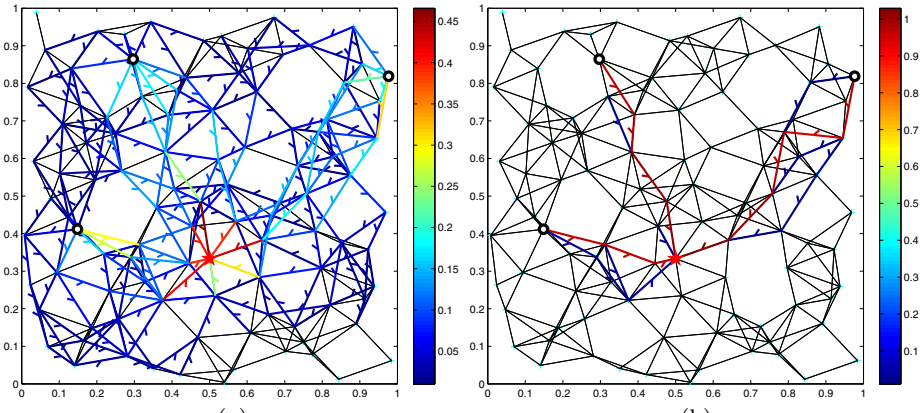


Fig. 4. Steady states obtained by QualNet for a 100-node network, including 1 sink (the star near the center) and 3 sources (the bold circles) with link color and width signifying the flux value under (a) $\mu = 0.5$ and (b) $\mu = 2$

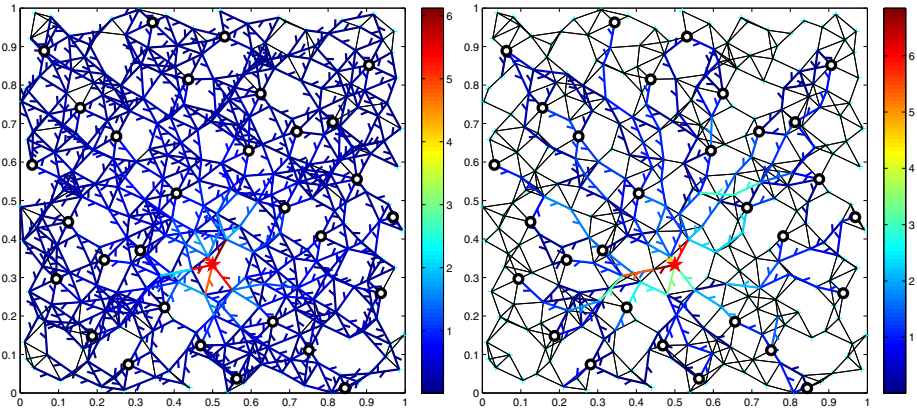


Fig. 5. Steady states obtained by QualNet for a 300-node network, including 1 sink (the star near the center) and 30 sources under (a) $\mu = 0.5$ and (b) $\mu = 2$

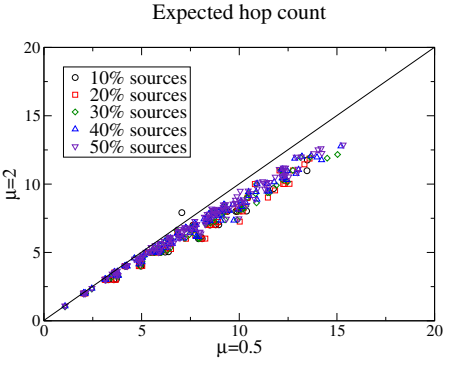


Fig. 6. Expected hop count for a 300-node network with 1 sink and 10% to 50% sources. Each point corresponds to two expected number of hops between one of the sources and the sink, with x-coordinate for $\mu = 0.5$ and y-coordinate for $\mu = 2$.

for balancing efficiency and robustness by determining whether a single route or multiple routes are created between sources and the sink.

Figure 6 reflects the efficiency of connectivity for a 300-node network with one sink and 10% to 50% of sources, with each data point signifying two expected hop counts between one source and the sink defined in (7) for $\mu = 0.5$ (x-coordinate) and $\mu = 2$ (y-coordinate). Although all points spread close to the diagonal, many lie below the diagonal towards the x-axis of $\mu = 0.5$, implying that the robustness provided by $\mu = 0.5$ comes at the cost of increased hop counts. Linear regression of the data shows that $\mu = 0.5$ requires approximately

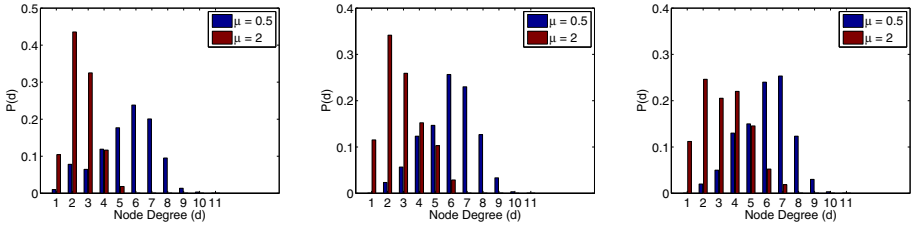


Fig. 7. Distribution of node degree for (a) 10%, (b) 30%, and (c) 50% sources on a 300-node network with 1 sink

10% more hops than $\mu = 2$. Table 1 presents the 1 and 2-fault tolerances for the same networks, showing that $\mu = 0.5$ achieves better robustness against node failure than $\mu = 2$, and that the network becomes less fault tolerant with more sources added. Figure 7 presents the distribution of node degree. As we would expect, $\mu = 0.5$ leads to a higher proportion of high degree nodes in a multi-route connectivity, whereas $\mu = 2$ leads to a greater proportion of low degree nodes in a single-route topology. The distinction blurs as more sources join the network.

6 Conclusion

In this paper, we design a slime mold inspired, self-organizing path formation protocol for WSNs. The localized protocol is effective in iteratively solving the underlying mathematical model which is globally coupled through the pressure field. The critical parameter μ controls whether single ($\mu > 1$) or multi-route ($\mu < 1$) connections emerge. We validate the protocol through linear stability analysis on a small model network, as well as via simulations on large realistic WSNs with one sink and varying numbers of sources. Future work will focus on extending the protocol to deal with real network traffic, exploring L_{ij} as a measure of link quality, and modifying the biologically inspired model to address WSNs with coordinated routing to multiple sinks.

Acknowledgements. This work is supported in part by National Science Foundation under grants CCF-0726556 and CNS-0347460.

References

1. Ben-Jacob, E., Cohen, I.: Cooperative organization of bacterial colonies: From genotype to morphotype. Annual Review of Microbiology 52, 779–806 (1998)
2. Li, K., Thomas, K., Rossi, L.F., Shen, C.C.: Slime-mold inspired protocol for wireless sensor networks. In: Proc. of the 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), pp. 319–328. IEEE Press, Los Alamitos (2008)

3. Nakagaki, T., Yamada, H., Toth, A.: Maze-solving by an amoeboid organism. Nature 407(6803), 470–470 (2000)
4. Scalable Network Technologies, Inc.: QualNet Simulator, http://www.scalable-networks.com
5. Stewart, P.A.: The organization of movement in slime mold plasmodia. In: Primitive Motile Systems in Cell Biology, pp. 69–78. Academic Press, London (1964)
6. Tero, A., Kobayashi, R., Nakagaki, T.: A mathematical model for adaptive transport network in path finding by true slime mold. J. Theor. Biol. 244, 553–564 (2007)
7. Tero, A., Takagi, S., Saigusa, T., Ito, K., Bebber, B.P., Fricker, M.D., Yumiki, K., Kobayashi, R., Nakagaki, T.: Rules for biologically inspired adaptive network design. Science 327, 439–442 (2010)

Solving the Multi-dimensional Multi-choice Knapsack Problem with the Help of Ants

Shahrear Iqbal, Md. Faizul Bari, and M. Sohel Rahman

AℓEDA Group
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka-1000, Bangladesh
{shahreariqbal,faizulbari,msrahman}@cse.buet.ac.bd

Abstract. In this paper, we have proposed two novel algorithms based on Ant Colony Optimization (ACO) for finding near-optimal solutions for the Multi-dimensional Multi-choice Knapsack Problem (MMKP). MMKP is a discrete optimization problem, which is a variant of the classical 0-1 Knapsack Problem and is also an NP-hard problem. Due to its high computational complexity, exact solutions of MMKP are not suitable for most real-time decision-making applications e.g. QoS and Admission Control for Adaptive Multimedia Systems, Service Level Agreement (SLA) etc. Although ACO algorithms are known to have scalability and slow convergence issues, here we have augmented the traditional ACO algorithm with a unique random local search, which not only produces near-optimal solutions but also greatly enhances convergence speed. A comparative analysis with other state-of-the-art heuristic algorithms based on public MMKP dataset shows that, in all cases our approaches outperform others. We have also shown that our algorithms find near optimal (within 3% of the optimal value) solutions within milliseconds, which makes our approach very attractive for large scale real time systems.

1 Introduction

The classical 0–1 Knapsack Problem (KP) is to pick up items for a knapsack to maximize the total profit, satisfying the constraint that, the total resource required does not exceed the resource constraint R of the knapsack. This problem and its variants are used in many resource management applications such as cargo loading, industrial production, menu planning, and resource allocation in multimedia servers [16]. The Multi-dimensional Multiple-choice Knapsack Problem (MMKP) is a variant of the classical 0–1 KP. Here we have n groups of items. Group i has ℓ_i items. Each item of the group has a particular value and it requires m resources. The objective of the MMKP is to pick exactly one item from each group for maximum total value of the collected items, subject to m resource constraints of the knapsack. In mathematical notation, let v_{ij} and $\overrightarrow{r_{ij}} = (r_{ij1}, r_{ij2}, \ldots, r_{ijm})$ be the value (profit) and required resource

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 312–323, 2010.

vector of the object o_{ij}, i.e., j-th item of the i-th group. Also assume that $\overrightarrow{R} = (R_1, R_2, \ldots, R_m)$ be the resource bound of the knapsack. Now, the problem is to

$$maximize \sum_{i=1}^{n} \sum_{j=1}^{\ell_i} x_{ij} v_{ij} \ (objective\ function),$$

$$subject\ to \sum_{i=1}^{n} \sum_{j=1}^{\ell_i} x_{ij} r_{ijk} \leq R_k \ (resource\ constraints)$$

where $k = 1, 2, \ldots, m$, $x_{ij} \in \{0, 1\}$ are the picking variables, and for all $i \in 1\ to\ n$, $\sum_{j=1}^{li} xij = 1$.

Fig 1 illustrates an MMKP. We have to pick exactly one item from each group. Each item has two resources, r_1 and r_2. Clearly we must satisfy $\sum(r_1$ of picked items$) \leq 17$ and $\sum(r_2$ of picked items$) \leq 15$ and maximize the total value of the picked items. Notably, it may happen that no set of items satisfying the resource constraints exists implying that no solution will be found.

In this paper, we have described two new algorithms for solving MMKPs. These algorithms are based on Ant Colony Optimization (ACO), which is a recently developed, population-based stochastic meta-heuristic [5,6]. ACO has been successfully applied to solve several NP-hard combinatorial optimization problems [4,21], such as traveling salesman problem [7,6], vehicle routing problem [9], and quadratic assignment problem [10,18].

This meta-heuristic belongs to the class of problem-solving strategies derived from nature. The ACO algorithm is basically a multi-agent system where low level interactions among the agents (i.e., artificial ants) result in a complex behavior of the whole ant colony. The basic idea of ACO is to model the problem under consideration as a searching problem, where a minimum cost path in a graph is searched; the artificial ants are employed to search for good paths. The pheromone trails are a kind of distributed information which is modified by the

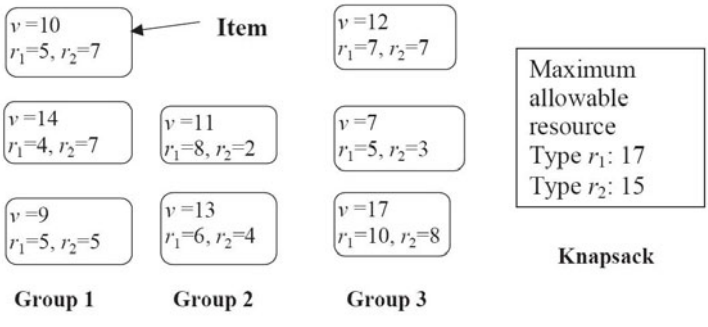


Fig. 1. Multi-dimensional Multiple-choice Knapsack Problem (figure borrowed from [1])

ants to reflect their experience accumulated during the problem solving. This substance influences the choices they make: the larger the amount of pheromone is on a particular path, the larger is the probability that an ant would select the path. Additionally these pheromone trails progressively decrease by evaporation. Intuitively, this indirect stigmergetic communication mean aims at giving information about the quality of path components in order to attract ants, in the following iterations, towards the corresponding areas of the search space.

MMKP has received significant amount of attention in the literature mostly motivated by capital budgeting, multimedia applications etc. There exist a number of heuristics in the literature for solving MMKP. Khan [16] proposed an algorithm named HEU, using the idea of aggregate resource consumption. In [15], a modified version of HEU named M-HEU was presented, which provides solutions with total value on average equal to 96% of the optimum. In [1] the authors presented a convex hull based heuristic called C-HEU, which is very fast and achieves optimality between 88% and 98%. Hifi et al. [12] proposed a guided local search-based heuristic and later improved upon it to achieve a "reactive" local search-based (RLS) algorithm [11]. Hernndez and Dimopoulos [20] also proposed a new heuristic for MMKP.

For solving MMKP with ACO, the most important design choice lies in deciding which component of the problem should be regarded as the pheromone depositing component. Here we have laid pheromone trails on each object selected in a solution. Essentially, the idea is to increase the desirability of each object selected in a feasible solution: during the constructing of a new solution, these objects will be more likely to be selected. The contributions of this paper are as follows.

We present two novel ACO based algorithms for solving MMKP. Both of these algorithms produce comparable results with the current state-of-the-art heuristic algorithms. To the best of our knowledge, this work is the first attempt to solve MMKP using ACO. An interesting aspect of our algorithm is the introduction of a novel and unique random local search algorithm for improving the solutions generated by the ant colony. This process, coupled with the natural behavior of the artificial ants produces near-optimal solutions and greatly enhances convergence speed of the ant colony.

The rest of the paper is organized as follows. Section 2 gives a brief description of ACO algorithms for solving the multi-dimensional knapsack problem (MKP), a related variant of KP. We present our main contribution in Section 3, where we describe our algorithms for solving MMKPs. Section 4 presents the experimental results along with an insightful discussion on the experimental results. Finally we briefly conclude in Section 5.

2 ACO and Multi-dimensional KP

As has already been mentioned we did not find any ACO based algorithm to solve MMKP in the literature. However there exist a number of ACO based solution for a more restricted variant of KP, namely MKP [17,8,2]. In MKP

resources have multiple dimensions as in MMKP; however there is no concept of group in MKP. As a result, MKP can be thought of as a restricted version of MMKP, which has all objects in a single group. The algorithms of [17,8,2] differ in deciding which component of the problem should be regarded as the pheromone depositing component and in the mechanisms of pheromone updating:

1. **Pheromone Trails on Each Object:** The first way is to lay pheromone trails on each object belonging to the current solution set [17]: the amount of pheromone represents the preference of the object.
2. **Pheromone Trails on Each Pair:** In this case, pheromone trails are laid on each pair (o_i, o_j) of successively selected objects of the solution set [8]: the idea is to increase the desirability of choosing object o_j when the last selected object is o_i.
3. **Pheromone Trails on All Pair:** The third one is to lay pheromone trails on all pairs of different objects of the solution set [2]. Here, the idea is to increase the desirability of choosing simultaneously two objects of S.
4. **Pheromone Diffusion Model:** The forth approach follows the same principle as the first one. Additionally it uses a pheromone diffusion scheme where pheromone trails are laid on objects that tend to occur together in previous solutions [14].

These approaches also differ in the way local heuristic information is defined. We are particularly interested in the dynamic local heuristic information used by [2,17,14] as defined below. Let S_k be the set of the selected objects at the k-th Iteration. For each candidate object j, the heuristic information $\eta_{S_k}(j)$ is given as follows:

$$\eta_{S_k}(j) \;=\; \frac{v_j}{\sum_{i\,=\,1}^{m} r_{ij}/d_{S_k}(i)} \tag{1}$$

where,

$$d_{S_k}(i) \;=\; R_i \;-\; \sum_{t \,\in\, S_k} r_{it} \tag{2}$$

Since S_k will be changed from step to step, the heuristic information is dynamic. we will be using a variation of above heuristic.

3 Description of the Proposed Algorithm

We have proposed two variation of the ACO algorithm for solving MMKPs namely ***AntMMKP-Random*** and ***AntMMKP-TopDb***. Both of the algorithm select groups randomly but the latter maintains a list of top k best solutions in order to direct the ants to a better area of the search space. They particularly follow the *MIN-MAX Ant System* [22], where explicit lower and upper bounds on pheromone values are imposed i.e. $\tau_{min} < \tau < \tau_{max}$, and all pheromone trails are initialized to τ_{max}. Below we describe these two algorithms in greater details.

3.1 Variation 1: AntMMKP-Random

This algorithm is described in Algorithm 1. At each cycle of this algorithm, k ants are used to build individual solutions. Each ant constructs a solution in a step by step manner. At first a group from the set of candidate groups is selected at random. All objects that violate resource constraints, are removed from this group. Then, the object with the highest probability (according to equation 5 below) is added to the solution. The probability of an object being selected depends on the amount of pheromone deposited on the object so far and its local heuristic value. The *candidategroups* data structure maintains a list of feasible candidate groups which can be considered next. After each ant has constructed a solution, the best solution of that iteration is identified and a random local search procedure and a random item swap procedure is applied to improve it. Then pheromone trail is updated according to the best solution. The algorithm stops either when an ant has found an optimal solution (when the optimal bound is known), or when a maximum number of cycles has been performed.

Algorithm 1. Algorithm AntMMKP-Random

```
Initialize pheromone trails to τ_max
repeat
  Solution S_globalbest ⇐ ∅
  for each ant k in 1...nants do
    Solution S_iterbest ⇐ ∅
    candidategroups ⇐ all the groups
    while candidategroups ≠ ∅ do
      C_g ⇐ Randomly select a group from candidategroups
      Candidates ⇐ {o_i ∈ objects in C_g that do not violate resource constraints}
      update local heuristic values
      Choose an object o_i ∈ Candidates with probability P_{S_k}(o_i)
      S_k ⇐ {S_k ∪ o_i}
      remove C_g from candidategroups
    end while
    if profit(S_k) > profit(S_iterbest) then
      S_iterbest ⇐ S_k
    end if
  end for
  S_iterbest ⇐ RandomLocalSearch(S_k)
  S_iterbest ⇐ RandomItemSwap(S_k)
  if profit(S_globalbest) < profit(S_iterbest) then
    S_globalbest ⇐ S_iterbest
  end if
  Update pheromone trails w.r.t S_iterbest
  if pheromone value is lower than τ_min then
    set pheromone ⇐ τ_min
  end if
  if pheromone value is greater than τ_max then
    set pheromone ⇐ τ_max
  end if
until maximum number of ycles reached or optimal solution found
```

Pheromone Trails. To solve MMKPs with ACO, the key point is to decide which components of the constructed solutions should be rewarded, and how to exploit these rewards when constructing new solutions. A solution of a MMKP is a set of selected objects $S = \{o_{ij} | x_{o_{ij}} = 1\}$ (i.e., an object o_{ij} is selected if

the corresponding decision variable x_{ij} has been set to 1). Given a constructed solution $S = \{o_{i_1 j_1}, \ldots, o_{i_n j_n}\}$, pheromone trails are laid on each objects selected in S. So pheromone trail τ_{ij} will be associated with object o_{ij}.

Pheromone Updating. Once each ant has constructed a solution, pheromone trails laying on the solution objects are updated according to the ACO metaheuristic. First, all amounts are decreased in order to simulate evaporation. This is done by multiplying the quantity of pheromone laying on each object by a pheromone persistence rate $(1 - \rho)$ such that $0 \leq \rho \leq 1$.

Then, pheromone is increased for all the objects in the best solution of the iteration. More precisely, let $S_{iterbest}$ be the best solution constructed during the current cycle. Then the quantity of pheromone increased for each object is determined by the function $G(S_{iterbest}) = Q.profit(S_{iterbest})$, where $Q = \frac{1}{\sum_{j=1}^{n} P_j}$ and $profit(S_{iterbest}) = \sum_{o_{ij} \in S_{iterbest}} v_{ij}$.

Algorithm 2. Algorithm for Random Local Search

```
procedure RandomLocalSearch(S)
Input: a solution S_k
Output: an improved solution S_k or input if no improvemnt found

    for a prespecified number of times do
      C_g ⇐ Randomly select a group
      for each object o_i ∈ C_g other than the one in S_k do
        S_tmp ⇐ include o_i removing the object selected in C_g
        if S_tmp not violates any resource constraints then
          if profit(S_k) < profit(S_tmp) then
            S_k ⇐ S_tmp
          end if
        end if
      end for
    end for
  return S_k
```

Heuristic Information. The heuristic factor $\eta_{S_k}(O_{ij})$ also depends on the whole set S_k of selected objects. Let $c_{S_k}(l) = \sum_{O_{ij} \in S_k} r_{ijl}$ be the consumed quantity of the resource l when the ant k has selected the set of objects S_k. And let $d_{S_k}(l) = R_l - c_{S_k}(l)$ be the remaining capacity of the resource l. We define the following ratio:

$$h_{S_k}(O_{ij}) = \sum_{l=1}^{m} r_{ijl}/d_{S_k}(l) \tag{3}$$

which represents the tightness of the object O_{ij} on the constraints l relatively to the constructed solution S_k. Thus, the lower this ratio is, the more the object is profitable. We integrate the profit of the object in this ratio to obtain a pseudo-utility factor. We can now define the heuristic factor formula as follows:

$$\eta_{S_k}(O_{ij}) = \frac{v_{ij}}{h_{S_k}(O_{ij})} \tag{4}$$

Constructing a Solution. When constructing a solution, an ant starts with an empty knapsack. At the k-th construction step ($k \geq 1$), an ant randomly selects a group and remove all the bad *Candidates* that violates resource constraints. It then updates the local heuristic information of the remaining candidate objects of the group and selects an object according to the following probability equation:

$$\rho_{S_k(O_{ij})} = \frac{[\tau_{S_k}(O_{ij})]^{\alpha}.[\eta_{S_k}(O_{ij})]^{\beta}}{\sum_{O_{ij} \in Candidates}[\tau_{S_k}(O_{ij})]^{\alpha}.[\eta_{S_k}(O_ij)]^{\beta}} \quad (5)$$

Here *Candidates* are all items from the currently selected group which do not violate any resource constraints. The construction process stops when exactly one item is chosen from each group.

Algorithm 3. Algorithm for Random Item Swap

```
procedure RandomItemSwap(S)
Input: a solution S_k
Output: an improved solution S_k or input if no improvemnt found

    for a prespecified number of times do
      for j = 1 to NUMBER-OF-ITEM-TO-FLIP do
        C_g ⇐ Randomly select a group
        O_i ⇐ Randomely select an item from C_g
        S_tmp ⇐ include o_i removing the object selected in C_g
      end for
      if S_tmp not violates any resource constraints then
        if profit(S_k) < profit(S_tmp) then
          S_k ⇐ S_tmp
        end if
      end if
    end for
  return S_k
```

Random Local Search. Random Local search described in Algorithm 2 is an exhaustive search within a group to improve the solution. It replaces current selected object of a group with every other object that do not violate resource constraints and checks if it is a better solution. The total procedure is repeated a number of times, each time for a random group.

Random Item Swap. Random Item Swap described in Algorithm 3 is an extended version of the random local search. In this case at a time, a specified number (> 1) of objects are swapped with other random objects from the same group without checking the resource constraints, then it checks if it is a valid solution and if it improves the solution.

3.2 Variation 2: AntMMKP-topdatabase

In this variation (Algorithm 4), the only difference from ***AntMMKP-Random*** is that, it maintains a database of top k solutions. After each iteration a small amount of pheromone is deposited in the pheromone trails of the objects belonging to the top k solutions. The motivation behind this strategy is to ensure quick convergence on good solutions and to explore better areas more thoroughly.

Algorithm 4. Algorithm AntMMKP-TopDb

```
Initialize pheromone trails to τ_max
topkdb ⇐ ∅ {data structure that holds topmost k solutions}
repeat
  Solution S_globalbest ⇐ ∅
  for each ant k in 1...nants do
    Solution S_iterbest ⇐ ∅
    candidategroups ⇐ all the groups
    while candidategroups ≠ ∅ do
      C_g ⇐ Randomly select a group from candidategroups
      Candidates ⇐ {o_i ∈ objects in C_g that do not violate resource constraints}
      update local heuristic values
      Choose an object o_i ∈ Candidates with probability P_{S_k}(o_i)
      S_k ⇐ {S_k ∪ o_i}
      remove C_g from candidategroups
    end while
    if profit(S_k) > profit(S_iterbest) then
      S_iterbest ⇐ S_k
    end if
  end for
  S_iterbest ⇐ RandomLocalSearch(S_k)
  S_iterbest ⇐ RandomItemSwap(S_k)
  if profit(S_globalbest) < profit(S_iterbest) then
    S_globalbest ⇐ S_iterbest
  end if
  update top database
  Update pheromone trails w.r.t S_iterbest
  Update pheromone trails w.r.t topdatabase
  if pheromone value is lower than τ_min then
    set pheromone ⇐ τ_min
  end if
  if pheromone value is greater than τ_max then
    set pheromone ⇐ τ_max
  end if
until maximum number of cycles reached or optimal solution found
```

Table 1. Solution Quality Comparison

Problem File	Exact	MOSER	HEU	CPCCP	RLS	FLTS	FanTabu	CCFT	Ant-R	Ant-T
I01	173	-	154	159	161	158	169	173	**173**	**173**
I02	364	294	354	312	354	351	354	352	**364**	**364**
I03	1602	1127	1518	1407	1496	1445	1557	1518	**1598**	**1600**
I04	3597	2906	3297	3322	3435	3350	3473	3419	**3562**	**3563**
I05	3905.7	1068.3	3894.5	3889.9	3847.3	3905.7	3905.7	3905.7	**3905.7**	**3905.7**
I06	4799.3	1999.5	4788.2	4723.1	4680.6	4793.2	4799.3	4799.3	**4799.3**	**4799.3**
I07	24587	20833	-	23237	23828	23547	23691	23739	**24170**	**24158**
I08	36877	31643	34338	35403	35685	35487	35684	35698	**36211**	**36246**
I09	49167	-	-	47154	47574	47107	47202	47491	**48204**	**48207**
I10	61437	-	-	58990	59361	59108	58964	59549	**60285**	**60300**
I11	73773	-	-	70685	71565	70549	70555	71651	**72240**	**72179**
I12	86071	-	-	82754	83314	82114	81833	83358	**84282**	**84251**
I13	98429	-	-	94465	95076	91551	94168	94874	**96343**	**96307**

4 Experimental Results

In this section, we assess the performance of the two algorithms, and compare them to other heuristic algorithms available in the literature. The datasets we used are the benchmark data of MMKPs from OR-library [3]. The algorithms were coded in java and run on a PC with intel core 2 duo 2.8 Ghz CPU, 2GB memory running Windows XP. The parameters are set as follows: $nants = 50$ (i.e., the number of ants is set to 50), $\alpha = 1$, $\beta = 5$, $\rho = 0.01$, $k = 10$ (for AntMMKP-TopDb), $\tau_{min} = 0.01$ and $\tau_{max} = 6$ times the amount each ant deposits if it selects an item. For random item swap we used four flip and run 1000 times, also in random local search the loop runs $n * 5$ times, where n is the number of groups.

Table 1 gives the comparison results of the performance of different algorithms including our two algorithms, namely, AntMMKP-Random (Ant-R) and AntMMKP-TopDb (Ant-T). For each instance, Table 1 reports the best solution found by MOSER [19], HEU [16], CPCCP [12], RLS [11], FLTS [13], FanTabu [13], CCFT [13] along with the exact solution reported in the data files and the best solutions of Ant-R and Ant-T found in 1000 runs. The results of the other algorithms were borrowed from [13]. Our algorithms clearly outperform all others on each file. Notably, for datasets I01, I02, I05 and I06 they found the exact solution.

If we compare the performance of the two algorithms, we see that Ant-T outperforms Ant-R for smaller data files. This clearly justifies our reasoning to maintain the database of top k solutions. The insight here is that where exploration area is comparatively small, thoroughly exploring better areas can give better solutions. But for large instances (file I10 and onward) the exploration area is much larger. So letting the ant colony explore more area rather than to converge to the better area so far seems preferable.

Figure 2 gives the time comparison of the two algorithms we developed. It reports the average time (milliseconds, over 20 runs) taken by each of our algorithm to reach within 3% of the known optimal solution for each of the instance file. Considering the solution quality, each of the algorithms run quite fast. Both of the algorithms give result before 1.5 seconds to reach within 3% of the optimal solution for data file I13 which is quite a large instance of MMKP consisting of 400 groups each having 10 objects and with number of resource dimension being 10. So our algorithms are attractive for large scale real time problems.

The random local search procedure presented in this paper improves the solution quality greatly in each iteration. In Figure 3 we have run three variation of Ant-R on instance file I07 with 100 groups, 10 items per group having resource dimension 10. At first we run the algorithm without the random local search. Then, we use a local search that we have developed earlier which tries to find a better object replacing the current selected object from all the groups in a order (not random). Finally the algorithm was executed with our random local search. Figure 3 clearly shows that both versions of the local search strategy are quite good for improving the solution, random local search being the better. From this comparison we can understand that the order of the selection of

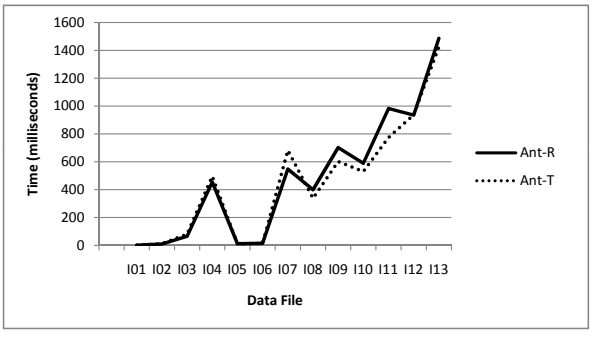


Fig. 2. Time comparison between Ant-R and Ant-T to reach within 3% of the optimal solution

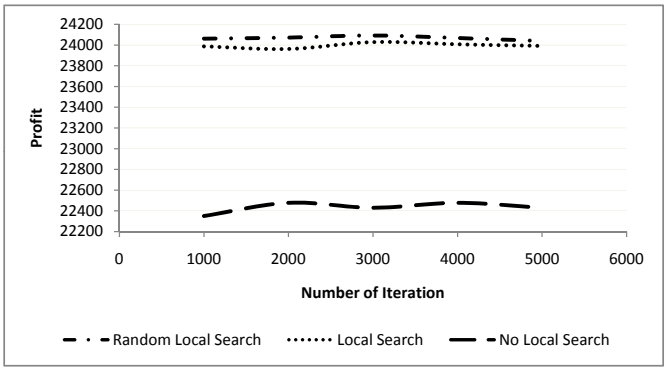


Fig. 3. Performance enhancement with our random local search

group while generating partial solution is very important to find good solutions for MMKP.

5 Conclusions

This paper is a first attempt to solve MMKPs using ant colony optimization. Here, we have proposed two new ACO algorithms for solving MMKPs along with a novel random local search strategy for performance improvement. We have presented simulation results, evaluating both runtime and solution quality of the proposed algorithms, and compared the solution quality of our algorithms with other existing state-of-the-art algorithms. From these simulation results it is clear that, our algorithms are the best in terms of solution quality and can also provide very fast near optimal solutions. The random local search seems to have provided the boost needed for providing such good quality solutions.

Acknowledgments. This research work was carried out as part of the M.sc. Engg. thesis of Shahrear Iqbal in the Department of Computer Science and Engneering, Bangladesh University of Engineering and Technology.

References

1. Akbar, M.M., Rahman, M.S., Kaykobad, M., Manning, E.G., Shoja, G.C.: Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. Comput. Oper. Res. 33(5), 1259–1273 (2006)
2. Alaya, I., Solnon, C., Ghèdira, K.: Ant algorithm for the multi-dimensional knapsack problem. In: International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2004), pp. 63–72 (2004)
3. Beasley, J.: OR-Library: Distributing test problems by electronic mail. The Journal of the Operational Research Society 41(11), 1069–1072 (1990)
4. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: From natural to artificial systems. J. Artificial Societies and Social Simulation 4(1) (2001)
5. Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristic: New ideas in optimization. McGraw-Hill Ltd., UK (1999)
6. Dorigo, M., Di Caro, G., Gambardella, L.M.: Ant algorithms for discrete optimization. Artificial Life 5(2), 137–172 (1999)
7. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evolutionary Computation 1(1), 53–66 (1997)
8. Fidanova, S.: Aco algorithm for mkp using different heuristic information. In: Dimov, I.T., Lirkov, I., Margenov, S., Zlatev, Z. (eds.) NMA 2002. LNCS, vol. 2542, pp. 438–444. Springer, Heidelberg (2003)
9. Gambardella, L.M., Taillard, É., Agazzi, G.: Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In: New Ideas in Optimization, pp. 63–76. McGraw-Hill, New York (1999)
10. Gambardella, L.M., Taillard, É., Dorigo, M.: Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society 50, 167–176 (1999)
11. Hifi, M., Michrafy, M., Sbihi, A.: A reactive local search-based algorithm for the multiple-choice multi-dimensional knapsack problem. Comput. Optim. Appl. 33(2-3), 271–285 (2006)
12. Hifi, M., Michrafy, M., Sbihi, A.: Heuristic algorithms for the multiple-choice multidimensional knapsack problem. Journal of the Operational Research Society 55, 1323–1332 (2004)
13. Hiremath, C.: New heuristic and metaheuristic approaches applied to the multiple-choice multidimensional knapsack problem. Ph.D. thesis, Wright State University (2008)
14. Ji, J., Huang, Z., Liu, C., Liu, X., Zhong, N.: An Ant Colony Optimization Algorithm for Solving the Multidimensional Knapsack Problems. In: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 10–16. IEEE Computer Society, Los Alamitos (2007)
15. Khan, S., Li, K.F., Manning, E.G., Akbar, M.M.: Solving the knapsack problem for adaptive multimedia systems. Studia Informatica Universalis 2, 157–178 (2003)
16. Khan, S.: Quality Adaptation in a Multisession Multimedia System: Model, Algorithms and Architecture. Ph.D. thesis, Department of Electrical and Computer Engineering, University of Victoria, ph.D. Dissertation (1998)

17. Leguizamon, G., Michalewicz, Z.: A new version of ant system for subset problems. In: Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999, vol. 2 (1999)
18. Maniezzo, V., Colorni, A.: The ant system applied to the quadratic assignment problem. IEEE Trans. on Knowl. and Data Eng. 11(5), 769–778 (1999)
19. Moser, M., Jokanovic, D., Shiratori, N.: An algorithm for the multidimensional multiple-choice knapsack problem. IEICE transactions on fundamentals of electronics, communications and computer sciences 80(3), 582–589 (1997)
20. Parra-Hernandez, R., Dimopoulos, N.J.: A new heuristic for solving the multichoice multidimensional knapsack problem. IEEE Transactions on Systems, Man, and Cybernetics, Part A 35(5), 708–717 (2005)
21. Parsons, S.: Ant colony optimization by marco dorigo and thomas stützle. Knowledge Eng. Review 20(1), 92–93 (2005)
22. Stützle, T., Hoos, H.H.: Max–min ant system. Future Generation Computer Systems 16, 889–914 (2000)

Theoretical Properties of Two ACO Approaches for the Traveling Salesman Problem

Timo Kötzing[1], Frank Neumann[1], Heiko Röglin[2], and Carsten Witt[3]

[1] Algorithms and Complexity, Max-Planck-Institut für Informatik, Saarbrücken, Germany
fne@mpi-inf.mpg.de
[2] Department of Quantitative Economics, Maastricht University, The Netherlands
[3] DTU Informatics, Technical University of Denmark, Kgs. Lyngby, Denmark

Abstract. Ant colony optimization (ACO) has been widely used for different combinatorial optimization problems. In this paper, we investigate ACO algorithms with respect to their runtime behavior for the traveling salesperson (TSP) problem. We present a new construction graph and show that it has a stronger local property than the given input graph which is often used for constructing solutions. Later on, we investigate ACO algorithms for both construction graphs on random instances and show that they achieve a good approximation in expected polynomial time.

1 Introduction

Stochastic search algorithms such as evolutionary algorithms (EAs) [3] and ant colony optimization (ACO) [2] are robust problem solvers that have found a wide range of applications in various problem domains. In contrast to many successful application of this kind of algorithms, the theoretical understanding lags far behind their practical success. Therefore, it is highly desirable to increase the theoretical understanding of these algorithms.

The goal of this paper is to contribute to the theoretical understanding of stochastic search algorithms by rigorous runtime analyses. Such studies have been successfully applied for evolutionary algorithms and have highly increased the theoretical foundation of this kind of algorithms. In the case of ACO algorithms the theoretical analyses of their runtime behavior has been started only recently [9,5,8,6]. We increase the theoretical understanding of ACO algorithms by investigating their runtime behavior on the well-known traveling salesperson (TSP) problem. For ACO the TSP problem is the first problem where this kind of algorithms has been applied. Therefore, it seems to be natural to study the behavior of ACO algorithms for the TSP problem from a theoretical point of view in a rigorous manner.

ACO algorithms are inspired by the behavior of ants to search for a shortest path between their nest and a common source of food. It has been observed that ants find such a path very quickly by using indirect communication via pheromones. This observed behavior is put into an algorithmic framework by

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 324–335, 2010.

considering artificial ants that construct solutions for a given problem by carrying out random walks on a so-called construction graph. The random walk (and the resulting solution) depends on pheromone values that are values on the edges of the construction graph. The probability of traversing a certain edge depends on its pheromone value.

One widely used construction procedure for tackling the TSP has already been analyzed in [11]. It constructs a tour in an *ordered* manner, where the iteratively chosen edges form a path at all times. In this paper, we give new runtime bounds for ACO algorithms using this construction procedure. On the other hand, we propose a new construction procedure, where, in each iteration, an *arbitrary* edge not creating a cycle or a vertex of degree 3 may be added to extend the partial tour. We analyze both construction methods and point out their differences.

Our analysis of these two ACO variants goes as follows. We first examine the locality of changes made, i.e., how many edges of the current-best solution are also in the newly sampled tour, and how many are *exchanged* for other edges. We then use these results as upper bounds on the time until certain desired local changes are made to derive upper bounds on the optimization time.

In particular, we show the following results:

- The ordered edge insertion algorithm exchanges an expected number of $\Omega(\log(n))$ many edges (Theorem 1) while the arbitrary edge insertion exchanges only an expected constant number of edges (Theorem 4).
- Arbitrary edge insertion has a probability of $\Theta(1/n^2)$ for any specific exchange of two edges (Corollary 1), while ordered edge insertion has one of $\Theta(1/n^3)$ [11].
- The simple TSP-instance analyzed in [11] is optimized by arbitrary edge insertion in an expected number of $O(n^3 \log(n))$ steps (Theorem 5), while the best known bound for ordered edge insertion is $O(n^6)$ ([11]).
- Both construction graphs lead in expected polynomial time to a good approximation on random instances.

The rest of the paper is organized as follows. In Section 2, we introduce the problem and the algorithms that are subject to investigations. We investigate the number of edge exchanges for large pheromone updates in Section 3 and prove runtime bounds for certain classes of instances in Section 4. Finally, we finish with some concluding remarks.

2 Problem and Algorithms

In this paper, we consider the symmetric Traveling Salesperson Problem (TSP). We are given a complete undirected graph $G = (V, E)$ and a weight function $w : E \to \mathbb{R}_+$ that assigns positive weights to the edges. The goal is to find a tour of minimum weight that visits every vertex exactly once and returns to the start vertex afterwards. We analyze an ACO algorithm called MMAS* (Min-Max Ant System – see Algorithm 1), already used in different theoretical studies [8,11].

MMAS* works iteratively, creating one new candidate solution x in each iteration, and keeping track of the best-so-far solution x^*. A new candidate solution for a target graph G is constructed by an artificial ant that performs a random walk on an underlying graph, called the *construction graph*, step by step choosing components of a new candidate solution. In this paper, we use edges of the given input as the components that influence this random walk. In each step of its random walk on the construction graph, we want the ant to choose an edge e in G with a probability based on *pheromone value* $\tau(e)$.[1] We use a procedure `construct` based on the pheromones τ as given in Algorithm 2. In this paper, we consider two different approaches of constructing new solutions by specifying the neighborhood function N of Algorithm 2 in Sections 2.1 and 2.2.

Algorithm 1. The algorithm MMAS*

```
1 function MMAS* on G = (V, E) is
2     τ(e) ← 1/|V|, for all e ∈ E;
3     x* ← construct(τ);
4     update(τ, x*);
5     while true do
6         x ← construct(τ);
7         if f(x) > f(x*) then
8             x* ← x;
9         τ ← update(τ, x*);
```

Algorithm 2. The algorithm `construct`

```
1 function construct based on τ is
2     for k = 0 to n − 2 do
3         R ← Σ_{y∈N(e_1,...,e_k)} τ(y);
4         Choose one neighbor e_{k+1} of e_k where the probability of selection of any
          fixed y ∈ N(e_1, ..., e_k) is τ(y)/R;
5     Let e_n be the (unique) edge completing the tour;
6     return (e_1, ..., e_n);
```

For each edge $e \in E$, the pheromones are kept within upper and lower bounds $\tau_{\max}$ and $\tau_{\min}$, respectively. The pheromone values change after each iteration of MMAS* according to a procedure `update` and an *evaporation factor* ρ: For a tour x, let $E(x)$ be the set of edges used in x; for each edge e, the pheromone values are updated such that the new pheromone values $\tau' = \texttt{update}(\tau, x)$ are such that

$$\tau'(e) = \begin{cases} \min\{(1-\rho)\cdot\tau(e)+\rho, \tau_{\max}\}, & \text{if } e \in E(x); \\ \max\{(1-\rho)\cdot\tau(e), \tau_{\min}\}, & \text{otherwise.} \end{cases}$$

[1] Note that, in this paper, we are not concerned with the use of *heuristic information*.

Here, ρ, $0 \leq \rho \leq 1$, is the evaporation factor which determines the strength of an update. As in [11], we use $\tau_{\min} = 1/n^2$ and $\tau_{\max} = 1 - 1/n$ throughout this paper, where n is the number of nodes of the input graph; further, initial values for pheromones are $1/n$. If in an iteration of MMAS* the pheromone values are such that, for exactly the edges of the best-so-far tour the pheromone values are at $\tau_{\max}$ and all others are at $\tau_{\min}$, we call the pheromones *saturated* at that iteration.

To measure the runtime of MMAS*, it is common to consider the number of constructed solutions. Often we investigate the expected number of constructed solutions until an optimal tour or a good approximation of an optimal tour is obtained.

2.1 The Input Graph as Construction Graph

To specify the construction graph, we need to introduce the neighorhood function N in Algorithm 2. The most common way of constructing a tour for TSP problem is to use the input graph as construction graph. A tour is constructed by having an ant start at some vertex, visit all vertices by moving to a neighbor of the current vertex, and finally coming back to the start vertex. We model this behavior with a neighbor set as follows. For each sequence σ of chosen edges, let $U(\sigma)$ be the set of unvisited nodes and $l(\sigma)$ the most recently visited node (or, if σ is empty, some distinguished node); let

$$N_{Ord}(\sigma) = \{\{l(\sigma), u\} \mid u \in U(\sigma)\}.$$

This set has the advantage of being easily computable and of size linear in the number of edges needed to complete the tour. We will discuss drawbacks of this neighborhood set later. We will refer to MMAS* using this neighborhood as MMAS^*_{Ord} ("Ord" is mnemonic for the "ordered" way in which edges are inserted into the new tour).

2.2 An Edge-Based Construction Graph

Alternatively, we can let the ant choose to add any edge to the set of edges chosen so far, as long as no cycle and no vertex are created. This is modeled by a neighbor set as follows. For each sequence σ of chosen edges, let $V(\sigma)$ be the set of previously chosen edges and

$$N_{Arb}(\sigma) = (E \setminus V(\sigma)) \setminus \{e' \in E \mid (V, \{e'_1, \ldots, e'_k, e'\}) \text{ contains a cycle or a vertex of degree} \geq 3\}.$$

This set has a size quadratic in the number of edges required to complete the tour. We will refer to MMAS* using this neighborhood as MMAS^*_{Arb} ("Arb" is mnemonic for the "arbitrary" way in which edges are inserted into the new tour).

3 Number of Edge Exchanges

In this section, we consider the expected number of edges that a newly constructed solution x differs from the best-so-far solution x^* if the pheromone values are saturated. In this case, the solution x^* can often be reproduced with constant probability and it is desirable that x^* and x only differ by a small (constant) number of edges. In such a situation, ACO algorithms are able to carry out improving steps by sampling solutions in their local neighborhood. In particular, for a tour t, we are interested in tours t' such that t and t' differ by exchanging 2 or 3 edges, called a 2-*Opt* or a 3-*Opt neighbor*, respectively.

3.1 The Behavior of MMAS$^*_{Ord}$

In the following we examine MMAS$^*_{Ord}$. We show that the expected number of edges where x^* and x differ is $\Omega(\log n)$. Thus, the MMAS$^*_{Ord}$ does *not* have the desired local property.

In the proof of the claimed result, we consider the following random process which captures the situation after an ant has left the high pheromone path for the first time. Let W be a walk on a sequence of t vertices. W starts at a random vertex, and will go to the just previous or following vertex in the sequence with equal probability, if both are available and unvisited. If only one is available and unvisited, W will go to this one. If none are available and unvisited, the walk will *jump* uniformly at random to an unvisited vertex. The walk ends as soon as all vertices are visited.

Lemma 1. *For each t, let X_t be the random variable denoting* number of jumps *made by the walk W on a path of t vertices. Then we have $\forall t \geq 3 : E(X_t) \geq \frac{1}{6}\ln(t)$.*

Proof. We start by giving a recursive definition of X_t. Clearly, $X_1 = 0$ and $X_2 = 0$. Let $t \geq 3$. The walk can start with uniform probability in any vertex, and will not jump if the first or last vertex has been chosen. Otherwise, with equal probability, the walk will start up or down. After visiting all nodes in the chosen direction, the walk will jump once, and then perform a walk according to X_i, where i is the number of unvisited nodes just before the jump. Thus, we get, for all $t \geq 3$,

$$\begin{aligned} E(X_t) &= \frac{1}{t}\sum_{i=2}^{t-1}\left(\frac{1}{2}(1 + E(X_{i-1})) + \frac{1}{2}(1 + E(X_{t-i}))\right) \\ &= \frac{t-2}{t} + \frac{1}{t}\left(\frac{1}{2}\sum_{i=2}^{t-1} E(X_{i-1}) + \frac{1}{2}\sum_{i=2}^{t-1} E(X_{t-i})\right) \\ &= \frac{t-2}{t} + \frac{1}{t}\sum_{i=1}^{t-2} E(X_i) = \frac{t-2}{t} + \frac{1}{t}\sum_{i=3}^{t-2} E(X_i) \end{aligned}$$

The claim is true for $t = 3$. We show the remainder of the claim of the lemma by induction on t. Let $t \geq 4$ and for all i, $3 \leq i < t$, $E(X_i) \geq \frac{1}{6}\ln(i)$. Using $t \geq 3$, we have $(t-2)/t \geq 1/3$. Thus, also using the induction hypothesis,

$$E(X_t) \geq \frac{1}{3} + \frac{1}{t}\sum_{i=3}^{t-2}\frac{1}{6}\ln(i)$$
$$= \frac{1}{3} + \frac{1}{t}\frac{1}{6}\ln(\prod_{i=3}^{t-2} i) = \frac{1}{3} + \frac{1}{t}\frac{1}{6}\ln((t-2)!/2) \geq \frac{1}{6}\ln(t).$$

□

Next we will give a lower bound on the expected number of edge exchanges which MMAS^*_{Ord} will make when saturated.

Theorem 1. *If in an iteration of $MMAS^*_{Ord}$ the pheromone values are saturated, then, in the next iteration of $MMAS^*_{Ord}$, the newly constructed tour will exchange an expected number of $\Omega(\log(n))$ of edges.*

Proof. It is easy to see that an ant leaves the path corresponding the currently best solution x^* with probability $\Omega(1)$ after having visited at most $n/2$ vertices. After the ant has left the path it performs on the remaining $r \geq n/2$ vertices as a walk similar to W on a path of length r. In fact, with constant probability, the ant will never leave the path again unless necessary, so that we get the result by applying Lemma 1. □

However, constructing new solutions with few exchanged edges is still somewhat likely. In [11] it is shown that the probability for a particular 2-Opt step is $\Omega(1/n^3)$. Taking a closer look at the analysis presented in this paper a matching upper bound on this probability can be extracted. In summary, we get the following result.

Theorem 2 ([11]). *Let t be a tour found by $MMAS^*_{Ord}$ and let t' be a tour which is a 2-Opt neighbor of t. Suppose that the pheromone values are saturated. Then $MMAS^*_{Ord}$ constructs t' in the next iteration of with probability $\Theta(1/n^3)$.*

3.2 The Behavior of MMAS^*_{Arb}

In this section we examine the expected number of edge exchanges of MMAS^*_{Arb}. In Theorem 4 we show that the expected number of edges where x^* and x differ is $\Theta(1)$. Thus, the MMAS^*_{Arb} *does* have the desired local property.

Theorem 3. *Let k be fixed. If in an iteration of $MMAS^*_{Arb}$ the pheromone values are such that, for exactly the edges of the best-so-far tour the pheromone values are at $\tau_{\max}$ and all others are at $\tau_{\min}$, then, in the next iteration of $MMAS^*_{Arb}$ with probability $\Theta(1)$, the newly constructed tour will choose k new edges and otherwise rechoose edges of the best-so-far tour as long as any are admissible.*

Proof. We call an edge with pheromone level $\tau_{\max}$ a "high" edge, the others are "low" edges. Let P be the set of all high edges (the edges of the best-so-far tour). We consider an iteration of MMAS^*_{Arb}. We analyze the situation where, out of

the n edges to be chosen to create a new tour, there are still i edges left to be chosen. In this situation, the edges chosen so far partition the graph into exactly i components. For each two components, there are between 1 and 4 edges to connect them (each component is a path with at most 2 endpoints, only the endpoints can be chosen for connecting with another component); thus, there are between $\binom{i}{2}$ and $\min\left(4\binom{i}{2}, \binom{n}{2}\right)$ edges left to be chosen. Further, when there are i edges left to be chosen for the tour, at most k of which are low edges, there are between i and $i+k$ high edges and between $\binom{i}{2} - (i+k)$ and $\min(4\binom{i}{2}, \binom{n}{2})$ low edges left to choose from.

For a fixed k-element subset M of $\{1, \dots, n\}$, and any choice of edges at positions M, we use the union bound to analyze the probability to rechoose as many other high edges as possible in all the other postions. This probability is lower bounded by

$$\begin{aligned} &1 - \sum_{i=1}^{n} \min\left(4\binom{i}{2}, \binom{n}{2}\right) \tau_{\min} \cdot \frac{1}{i\tau_{\max}} \\ &= 1 - \frac{\tau_{\min}}{\tau_{\max}} \left(\sum_{i=1}^{n/2} 4\binom{i}{2} \cdot \frac{1}{i} + \sum_{i=n/2+1}^{n} \binom{n}{2} \cdot \frac{1}{i} \right) \geq \frac{1}{4} > 0. \end{aligned}$$

For each k-element subset M of $\{1, \dots, n\}$, the probability of choosing a low edge on all positions of M, and choosing a high edge on all other positions is lower bounded by

$$\begin{aligned} &\frac{1}{4} \prod_{i \in M} \left(\binom{i}{2} - (i+k) \right) \tau_{\min} / ((i+k)\tau_{\max} + n^2 \tau_{\min}) \\ &\geq \frac{\tau_{\min}^k}{4} \prod_{i \in M} \left(\frac{i^2 - i}{2} - (i+k) \right) / (i+k+1) \geq \frac{\tau_{\min}^k}{4} \prod_{i \in M} \left(\frac{i}{2k+4} - 2 \right). \end{aligned}$$

Let $c_{i,k} = i/(2k+4) - 2$. Note that, for any set M with $|M| \leq k$, we have $\sum_{i=1, i \notin M}^{n} c_{i,k} = \Theta(n^2)$. Now we have that the probability of choosing low edges on *any* k positions is lower bounded by

$$\begin{aligned} &\frac{\tau_{\min}^k}{4} \sum_{\substack{M \subseteq \{1,\dots,n\} \\ |M|=k}} \prod_{i \in M} c_{i,k} \\ &= \frac{1}{4k! n^{2k}} \sum_{i_1=1}^{n} \left(\sum_{i_2=1, i_2 \notin \{i_1\}}^{n} \left(\dots \left(\sum_{i_k=1, i_k \notin \{i_1,\dots,i_{k-1}\}}^{n} \prod_{j=1}^{k} c_{i_j,k} \right) \right) \right) \\ &= \frac{1}{4k! n^{2k}} \left(\sum_{i_1=1}^{n} c_{i_1,k} \right) \left(\sum_{i_2=1, i_2 \notin \{i_1\}}^{n} c_{i_2,k} \right) \dots \left(\sum_{i_k=1, i_k \notin \{i_1,\dots,i_{k-1}\}}^{n} c_{i_k,k} \right) \\ &= \Theta(1). \end{aligned}$$

□

As a corollary to the proof just above, we get the following.

Theorem 4. *If in an iteration of $MMAS^*_{Arb}$ the pheromone values are such that, for exactly the edges of the best-so-far tour the pheromone values are at $\tau_{\max}$ and all others are at $\tau_{\min}$, then, in the next iteration of $MMAS^*_{Arb}$, the newly constructed tour will exchange an expected number of $O(1)$ of edges.*

As a further corollary to Theorem 3, we get the following.

Corollary 1. *Let t be a tour found by $MMAS^*_{Arb}$ and let t' be a tour which is a 2-Opt neighbor of t. Suppose that the pheromone values are such that for exactly the edges of t the pheromone values are at $\tau_{\max}$ and all others are at $\tau_{\min}$. Then $MMAS^*_{Arb}$ constructs t' in the next iteration with probability $\Theta(1/n^2)$.*

Proof. The tour t has $\Theta(n^2)$ many 2-Opt neighbors. By Theorem 3, MMAS^*_{Arb} will construct, with constant probability, a tour that exchanges one edge and otherwise rechooses edges of t as long as possible. This new tour is a 2-Opt neighbor of t. As all 2-Opt neighbors of t are constructed equiprobably (thanks to the symmetry of the construction procedure), we obtain the desired result. □

4 Runtime Bounds

4.1 A Simple Instance

An initial runtime analysis of ACO algorithms for the TSP problem has been carried out by Zhou in [11]. In that paper, the author investigates how ACO algorithms can obtain optimal solutions for some simple instances. The basic ideas behind these analyses is that ACO algorithms are able to imitate 2-Opt and 3-Opt operations.

A simple instance called G_1 in [11] consists of a single optimum, namely a Hamiltonian cycle where all edges have cost 1 (called light edges), while all remaining edges get a large weight of n (called heavy edges). The author shows that MMAS^*_{Ord} for arbitrary $\rho > 0$ obtains an optimal solution for G_1 in expected time $O(n^6 + (1/\rho)n \log n)$. The proof idea is as follows: As long as an optimal solution has not been obtained, there is always a 2-Opt or 3-Opt operation that leads to a better tour. Having derived a bound of $\Omega(1/n^5)$ for the probability of performing an improving 2- or 3-Opt step, the result follows since at most n improvements are possible and $O(\log n/\rho)$ is the so-called freezing time, i. e., the time to bring all pheromone values to upper or lower bounds.

In this section, we prove a bound of $O(n^3 \log n + (n \log n)/\rho)$ on the expected optimization time of MMAS^*_{Arb} for the instance G_1. This bound is considerably better than the $O(n^6)$ proved before in [11] for MMAS^*_{Ord}. At the same time, the analysis is much simpler and saves unnecessary case distinctions.

The following lemma concentrates on a single improvement. Following the notation in [11], let A_k, $k \leq n$, denote the set of all tours of total weight $n-k+kn$, i. e., the set of all tours consisting of exactly $n-k$ light and k heavy edges.

Lemma 2. *Let $\alpha = 1$ and $\beta = 0$, $\tau_{\min} = 1/n^2$ and $\tau_{\max} = 1 - 1/n$. Denote by X^t the best-so-far tour sequence produced by $MMAS^*_{Arb}$ on TSP instance G_1 until iteration $t > 0$ and assume that X_t is saturated. Then the probability of an improvement, given $1 \leq k \leq n$ heavy edges in X_t, satisfies $s_k = P(X^{t+1} \in A_{k-1} \cup \ldots \cup A_0 \mid X^t \in A_k) = \Omega(k/n^3)$.*

Proof. Consider an arbitrary light edge $e = \{u, v\} \notin T$ outside the best-so-far tour. Each vertex of G_1 is incident to 2 light edges, so both u and v are incident to exactly one light edge different from e. Since $e \notin T$, this implies the existence of two different heavy edges $e_0, e_1 \in T$ on the tour such that e_0 is incident on u and e_1 incident on v. Let $e'_0, e'_1 \in T$ with $e'_0 \neq e_0$ and $e'_1 \neq e_1$ be the other two edges on the tour that are incident to u and v, respectively. The aim is to form a new tour containing e and still e'_0 and e'_1 but no longer e_0 and e_1. Note that the set of edges $(T \cup \{e\}) \setminus \{e_0, e_1\}$ has cardinality $n - 1$ but might contain a cycle. If that is the case, there must be a heavy edge $e_2 \in T$ from the old tour on that cycle (since there is a unique cycle of light edges in G_1). Then we additionally demand that the new tour does not contain e_2. Since the undesired edges e_0, e_1 and possibly e_2 are heavy and e is a light edge outside the previous tour, any tour being a superset of $(T \cup \{e\}) \setminus \{e_0, e_1, e_2\}$ is an improvement compared to T.

For $1 \leq j \leq n/4$, we consider the following intersection of events, denoted by $M_e(j)$ and prove that $\text{Prob}(M_e(j)) = \Omega(1/n^4)$; later, a union over different j and e is taken to get an improved bound.

1. the first $j - 1$ steps of the construction procedure choose edges from $T^* := T \setminus \{e_0, e_1, e_2\}$ and the j-th step chooses e,
2. e'_0 is chosen before e_0 and e'_1 before e_1,
3. all steps except the first one choose from T^* as long as this set contains applicable edges.

Note that e_0 and e_1 are no longer applicable once $\{e, e'_0, e'_1\}$ is a subset of the new tour.

For the first subevent, assume that the first $i < j$ steps have already chosen exclusively from T^*. Then there $n-i$ edges from T and $n-i-3$ edges from T^* left. Finally, there are at most $n^2/2$ edges outside T. Using that X_t is saturated, the probability of choosing another edge from T^* is then at least $\frac{(n-i-3)\tau_{\max}}{(n-i)\tau_{\max}+n^2\tau_{\min}/2} \geq \frac{n-i-3}{n-i+1}$ (assuming $n \geq 2$). Altogether, the probability of only choosing from T^* in the first $j - 1$ steps is at least $\prod_{i=0}^{j-2} \frac{n-i-3}{n-i+1} \geq \left(\frac{3n/4-1}{3n/4+3}\right)^{n/4-1} = \Omega(1)$ since $j \leq n/4$. The probability of choosing e in the j-th step is at least at least $\tau_{\min}/n = 1/n^3$ since the total amount of pheromone in the system is at most n. Altogether, the first subevent has probability $\Omega(1/n^4)$.

The second subevent has probability at least $(1/2)^2 = 1/4$ since all applicable edges in T are chosen with the same probability (using that X_t is saturated).

For the third subevent, we study a step of the construction procedure where there are i applicable edges from T left and all edges chosen so far are from $T \cup \{e\}$. Now we need a more precise bound on the number of applicable outside T. Taking out $k \geq 1$ edges from T breaks the tour into k connected components,

each of which has at most two vertices of degree less than 2. Since $e \notin T$ has been chosen, at most two edges from T are excluded from our consideration. Altogether, the number of connected components in the considered step of the construction procedure is at most $i+2$, which means that there are at most $\binom{2(i+2)}{2} \leq 2(i+2)^2 \leq 18i^2$ edges outside T applicable. The probability of neither choosing e_2 nor an edge outside T in this situation is at least $\frac{i\tau_{\max}}{(i+1)\tau_{\max}+18i^2\tau_{\min}}$.

Hence, given the second subevent, the probability of the third subevent is at least

$$\begin{aligned}
\prod_{i=1}^{n-1} \frac{i \cdot \tau_{\max}}{(i+1)\tau_{\max} + 18i^2\tau_{\min}} &= \prod_{i=1}^{n-1} \left(\frac{i}{i+1} \cdot \frac{(i+1) \cdot \tau_{\max}}{(i+1)\tau_{\max} + 18i^2\tau_{\min}} \right) \\
&\geq \frac{1}{n} \prod_{i=1}^{n-1} \frac{i+1}{(i+1) + 18(i+1)^2/(\tau_{\max} \cdot n^2)} \geq \frac{1}{n} \left(\prod_{i=1}^{n} 1 + 18i/(\tau_{\max} \cdot n^2) \right)^{-1} \\
&\geq \frac{1}{n} \left(1 + \frac{18}{n-1} \right)^{-n} = \Omega(1/n),
\end{aligned}$$

altogether, the intersection $M_e(j)$ of the three subevents happens with probability $\Omega(1/n^4)$.

Finally, consider the union $M_e := \bigcup_{j \leq n/4} M_e(j)$, which refers to including e in any of the first $n/4$ steps. Since the $M_e(j)$ are disjoint for different j, we obtain $\text{Prob}(M_e) = (n/4) \cdot \Omega(1/n^4) = \Omega(1/n^3)$. Similarly, for all light edges $e \notin T$ (of which there are k), the events M_e are disjoint (as a different new edge is picked in the first step). Thus, the probability of an improvement is $\Omega(k/n^3)$ as desired. □

Theorem 5. *Let $\alpha = 1$ and $\beta = 0$, $\tau_{\min} = 1/n^2$ and $\tau_{\max} = 1 - 1/n$. Then the expected optimization time of $MMAS^*_{Arb}$ on G_1 is $O(n^3 \log n + n(\log n)/\rho)$.*

Proof. Using Lemma 2 and the bound $O(\log n/\rho)$ on the freezing time, the waiting time until a best-so-far solution with k heavy edges is improved is bounded by $O((\log n)/\rho) + s_k = O((\log n)/\rho + n^3/k)$. Summing up, we obtain a total expected optimization time of $O(n(\log n)/\rho) + \sum_{k=1}^{n}(1/s_k) = O(n^3 \log n + n(\log n)/\rho)$. □

4.2 Random Instances

The 2-Opt heuristic, which starts with an arbitrary tour and performs 2-Opt steps until a local optimum is reached, is known to perform well in practice in terms of running time and approximation ratio [7]. In contrast to this, it has been shown to have exponential running time in the worst case [4] and it has been shown that there are instances with local optima whose approximation ratio is $\Omega(\log n/\log\log n)$ [1]. To explain this discrepancy between theory and practice, 2-Opt has been analyzed in a more realistic model of random instances reminiscent of smoothed analyis [10]. In this model, n points are placed independently at random in the d-dimensional Euclidean space, where each point v_i $(i = 1, 2, \ldots, n)$ is chosen according to its own probability density

$f_i\colon [0,1]^d \to [0,\phi]$, for some parameter $\phi \geq 1$. It is assumed that these densities are chosen by an adversary, and hence, by adjusting the parameter ϕ, one can interpolate between worst and average case: If $\phi = 1$, there is only one valid choice for the densities and every point is chosen uniformly at random from the unit hypercube. The larger ϕ is, the more concentrated can the probability mass be and the closer is the analysis to a worst-case analysis. We analyze the expected running time and approximation ratio of MMAS^*_{Arb} and MMAS^*_{Ord} on random instances. For this, we have to take a closer look into the results from [4] which bound the expected number of 2-Opt steps until a good approximation has been achieved. We show the following theorem.

Theorem 6. *For $\rho = 1$, MMAS^*_{Arb} finds in time $O(n^{6+2/3} \cdot \phi^3)$ with probability $1 - o(1)$ a solution with approximation ratio $O(\sqrt[d]{\phi})$.*

Proof. As we have argued in Corollary 1, if all edges are saturated and there is an improving 2-Opt step possible, then this step is performed with probability at least $\Omega(1/n^2)$. From [4] we know that from any state, the expected number of 2-Opt steps until a tour is reached that is locally optimal for 2-Opt is at most $O(n^{4+1/3} \cdot \log(n\phi) \cdot \phi^{8/3})$ even if in between other changes are made to the tour that do not increase its length. Hence, using Markov's inequality we can conclude that MMAS^*_{Arb} has reached a local optimum after $O(n^{6+2/3} \cdot \phi^3)$ steps with probability $1 - o(1)$.

From [4], we also know that every locally optimal tour has an expected approximation ratio of $O(\sqrt[d]{\phi})$. Implicitly, the proof of this result also contains a tail bound showing that with probability $1 - o(1)$ every local optimum achieves an approximation ratio of $O(\sqrt[d]{\phi})$. The theorem follows by combining the previous observations and taking into account that for our choice of ρ all edges are saturated after the first iteration of MMAS^*_{Arb}. □

Taking into account that a specific 2-Opt operation in MMAS^*_{Ord} happens with probability of $\Omega(1/n^3)$ in the next step, we get the following results.

Theorem 7. *For $\rho = 1$, MMAS^*_{Ord} finds in time $O(n^{7+2/3} \cdot \phi^3)$ with probability $1 - o(1)$ a solution with approximation ratio $O(\sqrt[d]{\phi})$.*

5 Conclusions

Our theoretical results show that the usual construction procedure leads to solutions that are in expectation far away from the currently best one in terms of edge exchanges even if the pheromone values have touched their corresponding bounds. Due to this, we have examined a new construction graph with a stronger locality. On the other hand, this construction procedure has a high probability of carrying out a specific 2-opt operation which is important for successful stochastic search algorithms for the TSP problem. Afterwards, we have shown that both algorithms perform well on random instances if the pheromone update is high.

Acknowledgments. Timo Kötzing and Frank Neumann as well as Carsten Witt were supported by the Deutsche Forschungsgemeinschaft (DFG) under grants NE 1182/5-1 and WI 3552/1-1, respectively. Heiko Röglin was supported by a Veni grant from the Netherlands Organisation for Scientific Research (NWO).

References

1. Chandra, B., Karloff, H.J., Tovey, C.A.: New results on the old k-Opt algorithm for the traveling salesman problem. SIAM J. Comput. 28(6), 1998–2029 (1999)
2. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambrigde (2004)
3. Eiben, A., Smith, J.: Introduction to Evolutionary Computing, 2nd edn. Springer, Berlin (2007)
4. Englert, M., Röglin, H., Vöcking, B.: Worst case and probabilistic analysis of the 2-opt algorithm for the tsp: extended abstract. In: Bansal, N., Pruhs, K., Stein, C. (eds.) SODA, pp. 1295–1304. SIAM, Philadelphia (2007)
5. Gutjahr, W.J., Sebastiani, G.: Runtime analysis of ant colony optimization with best-so-far reinforcement. Methodology and Computing in Applied Probability 10, 409–433 (2008)
6. Horoba, C., Sudholt, D.: Running time analysis of ACO systems for shortest path problems. In: Stützle, T., Birattari, M., Hoos, H.H. (eds.) SLS 2009. LNCS, vol. 5752, pp. 76–91. Springer, Heidelberg (2009)
7. Johnson, D.S., McGeoch, L.A.: The traveling salesman problem: A case study in local optimization. In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization. Wiley, Chichester (1997)
8. Neumann, F., Sudholt, D., Witt, C.: Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. Swarm Intelligence 3(1), 35–68 (2009)
9. Neumann, F., Witt, C.: Runtime analysis of a simple ant colony optimization algorithm. Algorithmica 54(2), 243–255 (2009)
10. Spielman, D.A., Teng, S.H.: Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. J. ACM 51(3), 385–463 (2004)
11. Zhou, Y.: Runtime analysis of an ant colony optimization algorithm for TSP instances. IEEE Transactions on Evolutionary Computation 13(5), 1083–1092 (2009)

A Cooperative Network Game Efficiently Solved via an Ant Colony Optimization Approach

Pablo Romero, Franco Robledo, Pablo Rodríguez-Bocca,
Darío Padula, and María Elisa Bertinat

Facultad de Ingeniería, Universidad de la República, Uruguay
promero@fing.edu.uy

Abstract. In this paper, a Cooperative Network Game (CNG) is introduced. In this game, all players have the same goal: display a video content in real time, with no cuts and low buffering time. Inspired in cooperation and symmetry, all players should apply the same strategy, resulting in a fair play. For each strategy we shall define a score, and the search of the best one characterizes a Combinatorial Optimization Problem (COP). In this research we show that this search can be translated into a suitable Assymmetric Traveling Salesman Problem (ATSP). An Ant Colony Optimization (ACO) approach is defined, obtaining highly competitive solutions for the CNG. Finally, we play the game in a real context, using a new strategy in a Peer-to-Peer (P2P) platform, obtaining better results than previous strategies.

Keywords: COP, ATSP, ACO, P2P.

1 General Network Game

1.1 Definition

Consider a static network with $M > 1$ players and one server S who has an object, which all players are interested in. The server S cuts the object into very small pieces and, in each time slot, chooses at random only one player to send it. All players have a container that can allocate N pieces maximum, and every piece advance one position of the container in each time slot (the server always uses position 1 of the container, for only one benefited player). Each player can consult another in order to obtain a piece.

The consult works as explained next. Suppose that Player A consults Player B. Then chooses a permutation π of the natural subset $\{1, \ldots, N-1\}$, and checks if he has the piece at position $\pi(1)$. If not, checks B's container at position $\pi(1)$. If B has that piece, sends to A a copy of that piece, and the consult is successful. Otherwise, A repeats the procedure checking the container at position $\pi(2)$ and so on. Every consult finishes in a success (if the consulting player gets one piece) or in a fail (if after cheking the whole container, A does not get a piece). Players are awarded when they can fill position N of the container as many times as possible, but having at the same time the lowest number of pieces in the whole container. This will be mathematically defined next.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 336–343, 2010.

Each player $x \in \{1, \ldots, M\}$ chooses a permutation π_x. Be $p_x(i)$ the probability of filling position i of the container for Player x when the game is played unlimited in time (it is assumed that the number of pieces of the object is very large in relation with the container size N). Then, each player x is encouraged to choose a permutation in order to get the *continuity* $p_x(N)$ as high as possible and the expected number of pieces or *buffering time* $L = \sum_{i=1}^{N} p_x(i)$ as low as possible (see [12] for an alternative explanation in a network context).

2 Instructions for the CNG

The Cooperative Network Game (CNG) is a particular case of the General Network Game previously defined, and looks for a fair identical strategy (permutation) for all players. There is a Planner that can choose one permutation for each player. Given that he wants to have a fair and cooperative solution, he shall choose only one permutation π, which governs the consult between all players. In a stationary state, the probability of filling position i is the same for each player, named p_i. In [12] it is proved that the vector p_i complies that:

$$p_{i+1} = p_i + (1 - p_i)p_i s_i, \quad \forall i \in \{1, \ldots, N-1\} \ , \tag{1}$$

where s_i is *the strategic function* that depends on π. Specifically, for a given permutation π the strategic function s_i and the vector p_i comply the following Non-Linear System [3]:

$$(NLS(\pi)) \begin{cases} p_1 = \frac{1}{M}, & p_{i+1} = p_i + (1 - p_i)p_i s_i \quad \forall i \in \{1, \ldots, N-1\}; \\ s_{\pi_1} = 1 - \frac{1}{M}, & s_{\pi_{i+1}} = s_{\pi_i} + p_{\pi_i} - p_{\pi_i + 1} \quad \forall i \in \{1, \ldots, N-2\}. \end{cases}$$

The Non-Linear System $NLS(\pi)$ can be approximately solved for every particular π with the Newton-Raphson method, with quadratic convergence.

2.1 Score for the CNG

So far, we have not defined the objective for the CNG. There is a tradeoff between the continuity p_N and the buffering time $L = \sum_{i=1}^{N} p_i$, given that the vector p is monotonous increasing. The following analytical result will determine the score of the game:

Proposition 1. *Be X_π the random variable that counts the number of steps in a consult, needed to obtain a piece with the permutation strategy π. Then, its expected number is: $E(X_\pi) = \frac{M}{M-1} \sum_{i=1}^{N-1} \pi_i (p_{i+1} - p_i)$.*

Proof. See [3] for a detailed proof.

Thus, $E(X_\pi)$ is a linear combination of the jumps $p_{i+1} - p_i$. Moreover, it is monotonically increasing with the continuity p_N. Assuming that the whole consult is not longer than a time slot, it is convenient to maximize $E(X_\pi)$, which defines the score for the CNG:

Definition 1. *The score for the CNG for a permutation strategy π is the expected number of steps $E(X_\pi)$.*

Be P the space of all permutations of the natural subset $\{1, \ldots, N-1\}$. Consequently, in order to find the best strategy for the CNG, the next Combinatorial Optimization Problem (COP) must be solved:

$$(\text{COP}) \begin{cases} \max_{\pi \in P} E(X_\pi) \\ \text{s.t.} \quad p_i \textit{ complies with } \quad NLS(\pi). \end{cases}$$

3 Ideal Approach for the CNG

Lemma 1. ***Imperfect Continuity:*** $p_i < 1, \forall i \in \{1, \ldots, N\}$, $\pi \in P$

Proof. We know that $p_1 = \frac{1}{M} < 1$. Suppose now that $p_h < 1$ for some $h \in \{1, \ldots, N-1\}$. Then $p_{h+1} = p_h + (1 - p_h)p_h s_h < p_h + (1 - p_h) = 1$.

Lemma 2. ***Ascendent Occupation:*** $p_i < p_{i+1}, \forall i \in \{1, \ldots, N-1\}, \quad \forall \pi \in P.$

Proof. Trivial: by induction over the set $\{1, \ldots, N-1\}$.

Lemma 3. ***Descendent Composed****: s_{π_i} is strictly decreasing with i.*

Proof. Using Lemma 2: $s_{\pi_{i+1}} = s_{\pi_i} + p_{\pi_i} - p_{\pi_i+1} < s_{\pi_i}$.

Proposition 2. ***"Approximation Strategy Property" (ASP):***
Be $x = (x_1, \ldots, x_{N-1})$ an injective real-valued sequence. Then, there exists $\pi \in P$ that follows *the vector x in the next sense: If $x_i > x_j$ then $s_i > s_j$, $\quad \forall i, j \in \{1, \ldots, N-1\}$, where s is the strategic function associated with the permutation π.*

Proof. For every injective real-valued sequence x there exists a permutation of indices π such that $x_{\pi_1} > x_{\pi_2} > \ldots > x_{\pi_{N-1}}$. Using the Desendent Composed's Lemma, the strategic function s complies that $s_{\pi_1} > s_{\pi_2} > \ldots > s_{\pi_{N-1}}$. The result follows comparing the two previous inequalities.

The ASP permits to design a strategy whose occupation of the container is similar to a desired one. The ideal strategic function for a desired vector p can easily be obtained with (1):

$$s_{ideal_i} = \frac{p_{i+1} - p_i}{(1 - p_i)p_i}, \forall i \in \{1, \ldots, N-1\} \ . \tag{2}$$

We can approximate s_{ideal} via the ASP. The Follower System is illustrated in Fig. 1. Some ideal inputs were introduced into the Follower System. Here, we summarize the main experience (it is suggested to see [2]). This ideal approach *does not* reflect the behavior of chosen inputs. Rather, it is like a *dirty mirror*. However, the experience gained with chosen inputs permits to outstand a particular Subfamily of strategies defined next.

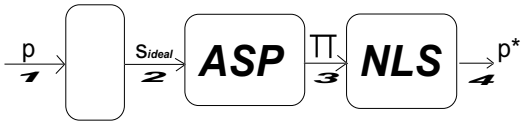


Fig. 1. Follower System: receives a desired probability of occupation p and returns a feasible occupation p^*, close to that occupation of the input. It permits to obtain the strategy π in Step 3, that achieves p^*.

Definition 2. *Subfamily of Strategies* $Sub(I, J)$
For each pair of naturals $(I, J) : I + J < N$*, we will call* $Sub(I, J)$ *to the subfamily of strategies that can be expressed in the next way:*

$$\pi(i) = N - i, \quad i = 1, \ldots, I, \quad \pi(I + j) = j, \quad j = 1, \ldots, J \ ;$$

$$\pi(I + J + k) = \left\lfloor \frac{N + J - I}{2} \right\rfloor + \left\lceil \frac{k}{2} \right\rceil (-1)^{k+1}, \quad k = 1, \ldots, N - I - J - 1 \ .$$

4 Feasible Approach Based on Ant Workers

From now on, we will attend the CNG trying to find the best strategy, on the lights of the score defined in the COP of Subsection 2.1. Considering high containers (e.g. $N > 15$) an exhaustive search for the best permutation results computationally prohibitive. In this section we will translate the COP into a suitable Assymmetric Traveling Salesman Problem (ATSP). This last is solved heuristically following an Ant Colony Optimization (ACO) approach, which is inspired in the way ants find the shortest path between their nests and their food [1]. The reader can find a deep analysis of this nature-inspired metaheuristic in [7,9,5,8].

Proposition 3. *"Translation of the Problem": An* $N-$*clique permits to obtain a bijection between a directed cycle that visits all its nodes and a permutation of* $\{1, \ldots, N - 1\}$.

Proof. Take any directed cycle $C = \{v_N, v_1, v_2, \ldots, v_{N-1}, v_N\}$.
Then, $\pi(i) = v_i$, $i = 1, \ldots, N - 1$ is clearly bijective.

Definition 3. *The function* $d(\pi, \pi^*)$ *(where* π *and* π^* *are permutations) counts the minimum number of swaps of elements to transform* π *into* π^*.

Proposition 4. (P, d) *is a metric space, being* P *the space of permutations.*

A Local Search can be defined substituting a permutation π with its best neighbor (which is at distance 1 from π). All these tools are used to define an ACO-based Algorithm, which finds high competitive strategies for the CNG:

```
Main Algorithm                              | Function: ApplyACO
1: d(E)   = Edges(ants)                     |1: Quality = Greedy
2: τ(E)   = Pheromones(Sub(I,J))            |2: FOR i = 1 TO ants
3: π      = ApplyACO(d,τ,iter,α,β,ρ)        |3:  π(i) = CycleACO(τ,Distances,α,β)
4: π_out = LocalSearch(π)                   |4:  τ    = NewPheromones(ρ,τ,Q,Q_max)
5: RETURN π_out                             |5: RETURN MostVisitedCycle(π(1),..,π(ants))
-------------------------------------------------------------------------------------
Function: Edges                               | Function: Pheromones
1: Distances = 1                              | 1: τ = 1
2: Quality   = Greedy                         | 2: Quality = Greedy
3: FOR i = 1 TO ants DO                       | 3: FOR EACH π ∈ Sub(I,J)
4:   π(i)     = VisitCycle(Distances)          | 4:   Q = Quality(π)
5:   Distances = UpdateCost(π(1),...π(i))     | 5:   τ = UpdateCost(π(1),..,π)
-------------------------------------------------------------------------------------
```

In the first stage of the Main Algorithm (Line 1), a non-negative assymmetric cost for each edge is initialized, with a learning mechanism based on ant exploration. The second block (Line 2) prepares the ACO application, via an initialization of the pheromones, which will permit to track cycles with high quality. The third block (Line 3) is the ACO application itself, which returns a strategy π. Finally, a local improvement is realized considering a typical local search.

The Function *Edges* translates the COP into an ATSP. To start, the cost of all edges are initialized to 1, and the Greedy strategy ($\pi_i = N - i$) is considered as a reference score. Then, each ant chooses probabilistically the next node to visit without making cycles. In the Function $VisitCycle$, ants choose the next step according with the next probabilities:

$$p(x_{j+1}) = \frac{Distances(x_j, x_{j+1})^{-1}}{\sum_{i \in NoCycle} Distances(x_j, x_i)^{-1}} \ . \tag{3}$$

So, shorter tours are preferable. Line 5 updates *Distances*. Function $UpdateCost$ finds the best strategy so far. Then, all edges in $\pi(i)$ are updated according with its score: $Distance(edge(j) \in \pi(i)) = 10(N-j) \times \frac{Q_{max}}{Q_{\pi(i)}}$, where Q_{max} is the best score obtained so far, and $edge(j)$ is the edge visited in order j in the cycle $\pi(i)$. There is an additional factor $N - j$, that avoids revisiting a cycle many times.

The pheromones for the immediate ACO application are initialized according with the experience obtained from the subfamily $Sub(I, J)$. Function *Pheromones* follows the same structure of *Edges*. The main difference between them is the deterministic cycles used in *Pheromones*, exploiting the high quality that provides the SubFamily $Sub(I, J)$ (see [2] for more details of the quality of $Sub(I, J)$). Function *ApplyACO* goes in parallel with a traditional ACO implementation, but each ant constructs one strategy. As a consequence, the exploration mechanism is slightly different. In Line 3, each ant makes a biased walk according with the probabilities:

$$p(x_{j+1}) = \frac{\tau(x_j, x_{j+1})^{\alpha} Distances(x_j, x_{j+1})^{-\beta}}{\sum_{i \in NoCycle} \tau(x_j, i)^{\alpha} Distances(x_j, i)^{-\beta}} \ , \tag{4}$$

where x_j is step j of the cycle, and α and β are the classical priority to pheromones and costs respectively. The updating of pheromones runs similar to classical implementations, based on an evaporation factor $\rho : 0 \leq \rho \leq 1$:

$$\tau(x_j, x_{j+1}) = (1-\rho)\tau(x_j, x_{j+1}) + \rho \times \frac{10(N-j)Q}{Q_{max}} , \quad (5)$$

where the factor $10(N-j)$ provides a similarity of magnitudes between pheromones and distances. *LocalSearch* chooses the best permutation among those which are at distance one from the output permutation of Function *ApplyACO*.

Theorem 1. *Be N the buffer size and $T(N)$ the average time for evaluating the quality $E(X_\pi)$. If we assume that the number of ants and the maximum number of iterations have order $O(N)$, then the average total time for running the Main Algorithm is $O(N^3T(N))$.*

Proof. *LocalSearch* dominates in computational effort, being C_2^{N-1} the number of neighbors for a given permutation. If the number of iterations is linear with N, the computational time for running *LocalSearch* is $O(N^3T(N))$.

5 Numerical Results

5.1 Comparison with Historical Strategies

There is an important reason to play the CNG: it is closely related with the delivery of live streaming contents in P2P networks. Basically, the players are now peers (computers) and the container is the storing capacity for each peer. Moreover, if the object to share is video, p_N is now the continuity of reproduction, and L is the buffering time. In P2P networks, two historical strategies for cooperation are the *Rarest First* $\pi_i = i$ and *Greedy* $\pi_i = N - i$ [12]. The former works properly in downloading, but not in streaming. The adaptation of ACO's parameters is based on [10], and then tuned in accordance with our particular problem. Our final implementation used the Main Algorithm with $\alpha = 0.4, \beta = 1.5, \rho = 0.5$ and 100 ants, for the common-network parameters $N = 40$ and $M = 100$. Fig. 2 shows that the obtained permutation achieves an excellent continuity and at the same time a latency comparable with the one reached by Greedy.

The obtained permutation was applied into a real platform named *GoalBit*, which is the first open-source P2P network that widely offers live video streaming

Strategies	Continuity	Latency
Rarest First	0.9571	21.0011
Greedy	0.9020	4.1094
Main Algorithm	0.9998	7.9821

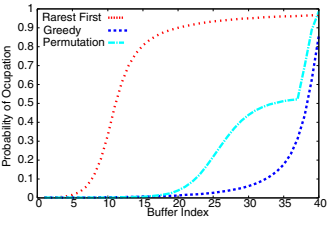


Fig. 2. Comparison between different strategies

to final Internet users [4,11]. GoalBit mantains the BitTorrents philosophy [6] considering the tit-for-tat strategy with optimistic unchoking, extending the success in the peer selection process. The clear weakness of BitTorrent for streaming applications is its peer selection strategy: Rarest First. The analysis realized in this paper shows its unacceptable latencies.

5.2 Results

Three strategies were considered to analyze their performance: Rarest First, Greedy and the Subfamily member $Sub(16, 1)$. The parameters $I = 16$ and $J = 1$ were tuned via an exhaustive search among all strategies $\pi \in Sub(I, J)$. The test case considers $N = 40$ and 45 peers (players) entering the network. Fig. 3 shows for each strategy and each peer, the initial buffering time, number of re-bufferings and mean buffering time, respectively. The Rarest First strategy has unacceptable start-up latencies for streaming purposes. On the other hand, the new permutation presents lower start-up latencies than Greedy for most of the peers. The interruption of the video signal is clearly higher for the Greedy strategy. Rarest First trades off latency for reduced cuts, but as seen before, has latencies in the order of minutes. Only four peers experimented longer cuts when our permutation strategy was applied. However the performance of the permutation strategy was higher in the rest of the peers, with respect to classical strategies.

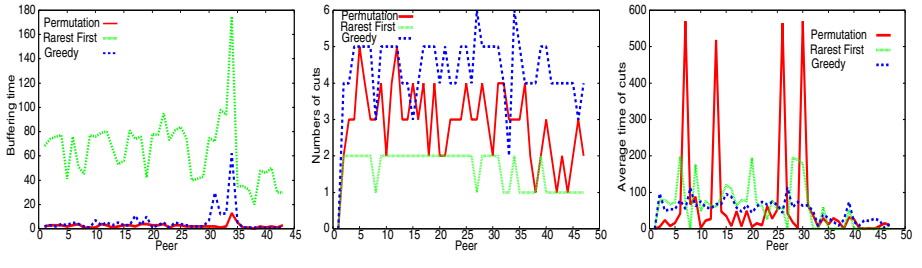


Fig. 3. Buffering time, Numbers of cuts and Average time of cuts when applying different strategies in GoalBit

6 Conclusions

An in-depth analysis for the CNG was presented, from both an ideal and feasible approaches. Although the ideal approach fails, it gives an insight for the design of competitive strategies. Feasible strategies were found via an Ant-Worker Algorithm, named the Main Algorithm. It translates an optimization problem into an assymmetric TSP, naturally explored via ACO. A Local Search phase finally improves ACO's output. The Main Algorithm was developed and tested with

common-network values of players and capacity ($M = 100$ and $N = 40$). Theoretically, it returned a strategy which achieves excellent continuity (very close to 1) and a reasonably low buffering time. Finally, the interest of playing the CNG could be verified when strategies were applied into a real P2P platform named GoalBit. A Subfamily member showed important advantages with respect to classical strategies. From both, theoretical and practical focuses, the results were highly competitive with respect to Greedy and Rarest First strategies.

References

1. Beckers, R., Deneubourg, J., Goss, S.: Trails and U-turns in the selection of the shortest path by the ant lasius niger. Journal of Theoretical Biology 159, 397–415 (1992)
2. Bertinat, M.E., Vera, D.D., Padula, D., Robledo, F., Rodríguez-Bocca, P., Romero, P.: Systematic procedure for improving continuity and latency on a p2p streaming protocol. To appear in Proceedings of IEEE Latin-American Conference on Communications (LatinCom 2009). IEEE, Los Alamitos (2009)
3. Bertinat, M.E., Vera, D.D., Padula, D., Robledo, F., Rodríguez-Bocca, P., Romero, P., Rubino, G.: A cop for cooperation in a p2p streaming protocol. To appear in Proceedings of International Conference in Ultra Modern Telecommunications (ICUMT 2009). IEEE, Los Alamitos (2009)
4. Bertinat, M.E., Vera, D.D., Padula, D., Robledo, F., Rodríguez-Bocca, P., Romero, P., Rubino, G.: Goalbit: The first free and open source peer-to-peer streaming network. To appear in Proceedings of the 5th international IFIP/ACM Latin American conference on Networking, pp. 49–59. ACM, New York (2009)
5. Blum, C.: Ant colony optimization: Introduction and recent trends. Physics of life Reviews 2, 353–373 (2005)
6. Cohen, B.: Incentives build robustness in bittorrent, vol. 1, pp. 1–5 (May 2003), www.bramcohen.com
7. Dorigo, M., Birattari, M., Stützle, T.: Artificial ants as a computational intelligence technique. Tech. Rep. 23, Institut de Recherches Interdisciplinaires, Université Libre de Bruxelles (September 2006)
8. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
9. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
10. Duan, H., Ma, G., Liu, S.: Experimental study of the adjustable parameters in basic ant colony optimization algorithm. IEEE Congress on Evolutionary Computation 1(1), 149–156 (2007)
11. GoalBit - The First Free and Open Source Peer-to-Peer Streaming Network (2008), http://goalbit.sf.net/
12. Zhou, Y., Chiu, D.M., Lui, J.: A Simple Model for Analyzing P2P Streaming Protocols. In: Proceeding of the IEEE International Conference on Network Protocols (ICNP 2007), Beijing, China, pp. 226–235 (October 2007)

A Deterministic Metaheuristic Approach Using "Logistic Ants" for Combinatorial Optimization

Rodolphe Charrier, Christine Bourjot, and François Charpillet

LORIA, Campus Scientifique,
Vandoeuvre-lès-Nancy, France
rodolphe.charrier@loria.fr

Abstract. Ant algorithms are usually derived from a stochastic modeling based on some specific probability laws. We consider in this paper a full deterministic model of "logistic ants" which uses chaotic maps to govern the behavior of the artificial ants. We illustrate and test this approach on a TSP instance, and compare the results with the original Ant System algorithm. This change of paradigm —deterministic versus stochastic— implies a novel view of the internal mechanisms involved during the searching and optimizing process of ants.

Keywords: Metaheuristics, Chaotic Map, Optimization, Swarm Intelligence, Ant Algorithm.

1 Introduction

Ant algorithms constitute a family of stochastic models mainly based on the following probability function used by artificial ants as a decision function:

$$p_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l \in \mathcal{N}_i}(\tau_{il})^{\alpha}(\eta_{il})^{\beta}} \tag{1}$$

where τ_{ij} denotes the amount of pheromone on the edge (i, j) linking node i to node j, and η_{ij} denotes an extra heuristics adapted to the problem[1], and $\mathcal{N}_i$ is the set of other existing edges from node i. The exponents α and β are greater or equal to 1, but commonly equals 2 to get the best performances. This probability law derives from the law found by Deneubourg to fit statistically with the experimental data of the famous "double bridge experiment" [4] involving only two edges. This stochastic decision enables a colony of agents (artificial ants) to find or approximate good solutions for many hard optimisation problems [5].

This paper focuses on the stochastic foundations of the ant algorithm modeling: more precisely our concern is with the existence of an alternative deterministic model which would exhibit the dynamical aspects of the involved processes. Some works have been published in this way of modeling, by requiring the hypothesis of chaotic dynamics in ant behaviors [2,6].

[1] The inverse of the edge distance in the TSP case.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 344–351, 2010.

In the same way, we propose the deterministic model of "logistic ants" in this paper. This model is inspired by some theoretical studies on iterated nonlinear maps, namely logistic maps or quadratic maps, which are well known to produce chaotic time series [3]. This feature is needed for simulating stochastic behaviors. The main advantage of this approach lies in the possibility of controlling the chaotic properties of the iterated map through a single parameter in our case. We deal therefore with the deterministic chaos theory, to "replace" the probability theory. This field provides tools like bifurcation diagrams to monitor the processes. The presentation of the logistic ant model constitutes the first section of this paper.

The logistic ant model has been already applied on the binary bridge experiment and has proved to simulate the symmetry breaking of the problem. It has moreover produced the same shape of data series as the experimental ones [1]. But it has never been applied to optimization problems, contrary to ant algorithms. We intend therefore to validate our approach by comparing the logistic ant model to the "Ant System" algorithm [5] —one of the first ant algorithm instance in the family— on a Travelling Salesman Problem (TSP) benchmark with 48 nodes. The objective is at this stage to prove that the concept is relevant for optimization, not to deal with a hard TSP. This constitutes the second section of this paper. The results we get are encouraging and our last section is dedicated to discuss these results.

2 The Logistic Ant Model for TSP

This section is devoted to the design of the logistic ant system. In fact this system is a reactive multi-agent system (MAS) composed of an environment plus many (logistic) agents. The agents interact through the environment by a pheromone field, that's why we call them "ants". However, the difference of logistic agents compared to stochastic ants lies in the decision process of logistic ants which is governed by a deterministic logistic map, in contrary to the stochastic law (1).

2.1 Metaheuristic Principles of Ant Algorithms

Let us specify this in the case of a TSP problem where the objective is to find the shortest Hamiltonian cycle in a weighted graph[2]. The generic considered graph is denoted $G = (V, E)$ where V is the set of $|V| = n$ vertices and E the set of edges. In our case the graph is symmetric and has $\frac{n(n-1)}{2}$ edges.

Before describing the different parts of the logistic ant system, let us recall some rooting principles of ant algorithms. We use the same global metaheuristic method using a pheromone field to perform optimization on a symmetric TSP, that is:

- N ants forming a colony are involved at the same time on a given TSP,
- the algorithm proceeds in a global loop composed of global steps,

[2] We will only consider symmetric TSP in this paper.

– during a global step, each ant achieves individually an hamiltonian cycle from a random initial position and marks it by an amount of pheromone,
– each global step ends with a pheromone reinforcement of the best cycle, when all ants have finished their cycle.

In this paper, the elementary discrete time step t corresponds to the process time needed to achieve a "local loop", that is a loop where all ants in the colony has moved into a new vertex. A global step lasts therefore $T = n-1$ time steps for all ants to cover an Hamiltonian cycle in parallel. We keep the naming of local loop relative to the vertex and global loop relative to the graph, to distinguish the different levels of schedulling in the algorithm. The global loop lasts until a fixed limit of time steps is reached: this is the criterion to stop the algorithm in this study. The best optimization performance among the colony is then recorded.

2.2 The Environment Design

We use the concept of environment of the MAS paradigm to include all the data related to the graph problem. In this way, our MAS environment is composed of a basic geometric space denoted E and many fields on it. E corresponds here to the $2D$ discrete space $\mathbb{N}_n^* \times \mathbb{N}_n^*$ where $\mathbb{N}_n^* = \{1, \cdots, n\}$ and n is the number of vertices in the considered graph.

The Notion of Field. The field concept is the means we use to structure the data in the environment and to describe dynamically all the processes within the logistic ant system. A field is defined as a mapping between the geometric space E and the real set. It may also be seen as a data layer of the environment, but remains mathematically a function. We distinguish the fields notably by the origine of their data: an endogenous field comprises data produced by agents, whereas an exogenous field comprises external data, set by the problem. On the other hand a field can be dependent on time or not. In all the following, (k, l) denotes a pair of coordinates in the space E.

Main Fields. The environment is composed of the following main fields:

– The adjacency field $\mathcal{A}$ corresponds simply to the connectivity matrix of the graph and is an exogenous field.
– The weight field stores the exogenous data relative to the given distances in the graph. Here we use a formula to transform all distances into the interval $[0, 1]$. In this way, we intend to have a generic design, scale-free and independent of the units used:

$$\mathcal{W}(k, l) = \frac{\min_{(i,j)\in E} d(i, j)}{d(k, l)} \tag{2}$$

where $d(i, j)$ gives the distance of the edge (i, j).
– The pheromone field $\mathcal{T}^t$ is dynamically build by the ant colony. It is therefore a cumulative endogenous field: The pheromone field is initialized to 0. It is characterized by a cumulative updating process and an evaporation coefficient ρ.

Other Fields. Other fields are needed in our environment modeling:

- A field of visited vertices and edges denoted by $\mathcal{H}_i^t$ stores, for an ant i at each time t during a global step of the algorithm, the taboo list of edges and nodes already visited by the ant.
- A field of ordered edges $\mathcal{O}^t$ maintains a list of edges related to each node of the graph, ordered by the amount of pheromone. This field is needed because of the determinitic nature of the choosing decision process of logistic ants (cf section 2.4).
- A field of influence: the pheromone field is a dynamical field, and it has to be updated after each ant has achieved a hamiltonian cycle. That is why we involve also an "influence" pheromone field denoted $\widetilde{\mathcal{T}}$, which is a temporary field for the global update of the pheromone field.

2.3 Design of the Logistic Ant

Rooting Principles

Internal state definition of the logistic ant. Let us describe now the internal behavior of the logistic ant. The logistic ant i is a reactive agent with the internal state $s_i^t = \langle x_i^t, a_i^t \rangle$ at time step t:

- $x_i \in [0, 1]$ is the decision variable of the agent,
- $a_i \in [0, 1]$ is the internal control variable of the agent which governs its type of dynamics.

The interpretation of the internal variables becomes clear within the decision function of the agent, which is a conjugate form of the logistic map:

$$f(x, a) = 1 - a(2\ x - 1)^2 \tag{3}$$

Here, both x and a belong to the interval $[0, 1]$: x is the main variable and a the control parameter of the map. When the a value is set, one can iterate this function according to the recurrence $x^{t+1} = f(x^t, a)$. Numerical studies on this recurrence for many iterations (about hundreds) lead to three type of results: a fixed point or a periodic cycle or chaotic (aperiodic) series. The asymptotic dynamical properties of this map are summed up in the bifurcation diagram (1).
Parametric control for exploration and exploitation. The basic idea of the algorithm consists in using the dynamical properties of the logistic map by modifying dynamically the a value as the algorithm runs and in respect to the optimisation objective. Chaos occurs within the right part of the diagram (1) which may correspond to the exploration phase, whereas fixed points occur in the left part which may correspond to the exploitation phase. This modulation of a is achieved by the perception and action functions/operators. Our algorithm at the local level of ants will lead from an exploration phase to an exploitation phase, that is from a high a-value ($a \simeq 1$) to a low a-value ($a \simeq 0$).

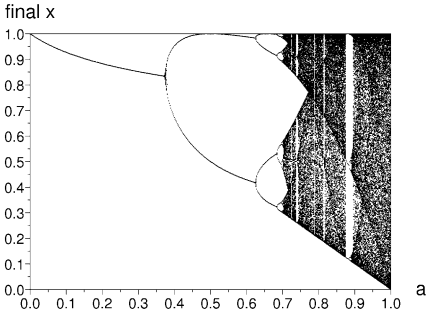


Fig. 1. Bifurcation diagram of the iterated map $x^{t+1} = 1 - a(2\ x^t - 1)^2$ with 500 loops

2.4 Inside the Logistic Ant

The internal processing of a logistic ant follows a sensorimotor scheduling — typical of a cybernetics approach—, that is a perception-decision-action process. This scheduling is achieved during an elementary time step of the algorithm, in parallel by each ant of the colony.

The Perception Process. Let us consider an ant i on a given vertex k of the graph, let V_k denote the set of all vertices connected with vertex k and not yet visited (not belonging thus to the taboo list). The perception operator acts on the pheromone field according to the formula:

$$P_i(k) = max_{l \in V_k}\{\mathcal{T}(k,l)\} \tag{4}$$

This perception returns simply the maximum amount of pheromone from a given vertex.

The Decision Process. The decision process performs the transition of the internal state of the logistic ant. For an ant i, it is formalized as a dynamical system between two time steps t and $t+1$:

$$\begin{cases} a_i^{t+1} = \dfrac{1}{1 + e^{\alpha\ (P_i^t(k) - \tau_0)}} \\ \\ x_i^{t+1} = f(x_i^t, a_i^{t+1}) \end{cases} \tag{5}$$

The updating of a_i regulates the adaptation behavior of ants by means of a sigmoid function which fixes the envelop of the decreasing variation of the control variable in function of perceptions.

The Action Process. After updating the decision variable, the logistic ant effects local actions:

- Choose an edge among the ordered edges list (through the field $\mathcal{O}$) from the current vertex in proportion to the value of the decision variable x and move on the choosen vertex, denoted l^*.
- Update the set of visited edges and vertices, that is the field $\mathcal{H}$.
- Update the influence pheromone field by the following formula:

$$\widetilde{\mathcal{T}}_i^{t+1}(k, l^*) = x_i^{t+1}\ \mathcal{W}(k, l^*) \tag{6}$$

2.5 Reaction of the Environment

The environment reactions occur once at the end of each time step when the N ant local loop are made up, once at the end of each global step to reinforce the best cycle.

The reaction process at each time step consists in the updating of the field $\mathcal{O}$, the field $\mathcal{H}_i$ for each ant, and the global pheromone field according to:

$$\mathcal{T}^{t+1} = \mathcal{T}^t + \sum_{i=1}^{N} \widetilde{\mathcal{T}}_i \tag{7}$$

At the level of a global step, the reaction consists in the reinforcement of the pheromone field for the best cycle among the colony. Let L_i denote the distance of the cycle for an ant i and G_{min} denote an inferior bound of the minimal distance of a cycle defined by the sum of minimal distances from all edges. The amount of pheromone $\triangle\tau \in [0, 1]$ for reinforcing the best cycle is given by:

$$\triangle\tau = \frac{G_{min}}{\min_i\{L_i\}} \tag{8}$$

This formula is independent of the distance units used and does not depend on any parameter. The updating of the pheromone field according to the evaporation coefficient ρ is then very similar as the Ant system algorithm. Finally the reaction process resets the ordered edges field, the visited edges field, and the influence pheromone field to their initial values.

3 Simulation and Results of the Logistic Ant Algorithm

Simulations have been performed on 20000 elementary time steps on the "att48" TSP instance from the TSPLIB library for wich the optimal cycle equals 10628. Different ant number N and evaporation coefficient $\rho \in \{0, 0.01, 0.02, \cdots, 0.1\}$ have been tested. We have compared the best results between the classical Ant System algorithm and our logistic ant algorithm on 5 runs for each initial configuration. The comparison of the best results is given in table 1. The sigmoid function used by logistic ants has been fixed to the following parameter: $\alpha = 0,04$ and $\tau_0 = 30,0$. In terms of computation time, the running time of the logistic ant algorithm is in average twice the runnig time of the Ant System algorithm with the same implementation conditions. Both algorithms failed in finding the optimal solution in the limited laps of time. Some points have to be mentioned:

Table 1. Comparison of best results between the "Ant System" (AS) algorithm and the Logistic Ant algorithm (LA). Each cell gives a pair (ρ_{best}, L_{best}).

Number of agents	10	20	30	40
AS	(0.0, 11028)	(0.06, 10845)	(0.06, 10847)	(0.08, 10777)
LA	(0.01, 11074)	(0.0, 11049)	(0.02, 10894)	(0.01, 11026)

- best results are very close between both algorithms,
- the AS algorithm converges very fast towards good solutions, whereas it takes more time for the LA algorithm to find good solutions,
- the increase of the ant number has low impact on the results in the AS case, whereas it speeds up the convergence in the LA case,
- the evaporation coefficient has to be near 0.1 to give the best results in the AS case, whereas much lower values are needed in the LA case.

4 Discussion and Interpretation of the Logistic Ant Algorithm

We consider that the performances of the logistic ant system are convincing to carry on this model for optimization. We foresee some advantages of this approach:

- the deterministic nature of the logistic ant model enables to make clear the underlying mechanisms involved in the ant colony. We have shown indeed that the pheromone field modifies the internal control variable of agents and impacts its future decisions through the nonlinear logistic map. The pheromone field reveals to be both a control field in dynamical terms and a decision field.
- This model constitutes therefore a metaheuristic approach, because of the genericity of the convergence principle: the convergence profile is given for each ant by the bifurcation diagram of the logistic map. This convergence principle is an abstract one and can be adapted to many types of optimization functions.
- The logistic ant model may be linked with approaches involving biological chaos as it is mentionned in the introduction of this paper.

The main drawback we may see in this model lies in the theoretical impossibility to prove at the present time the asymptotic convergence to the optimal solution, instead of the ant colony algorithm family. By contrast the convergence process may be known within each ant through the generic bifurcation diagram of the logistic map and according to the variation of the ant control variable. But it does not lead to know the convergence process at the global level. We intend to compensate for this drawback by keeping several chaotic agents in the environment to maintain an exploring level.

5 Conclusion

We have shown in this paper a way to build a full deterministic model of ant algorithm by means of logistic maps as decision functions. The results obtained by our "logistic ants" reveal to be comparable to the Ant System algorithm on the same TSP instance. The main advantage of this deterministic model lies in the mechanism principles it involves: the optimization process results from an internal parametric control of ants on the logistic map through the field of pheromone. An other interesting aspects of the model lies in the systemic formalization using a field-based environment and a sensori-motor loop in ants, which is generic and independent of the implementation on computers. It enables therefore to be applied to many types of problems. Our assumption is finally that the logistic ant model may be linked with realistic biological chaotic phenomena.

References

1. Charrier, R., Bourjot, C., Charpillet, F.: A nonlinear multi-agent system designed for swarm intelligence: The logistic MAS. In: International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2007, Boston (2007)
2. Cole, B.J.: Is animal behaviour chaotic? Evidence from the activity of ants. Proceedings of the Royal Society: Biological Sciences 244(1311), 253–259 (1991)
3. Collet, P., Eckmann, J.P.: Iterated Maps on the Interval as Dynamical System. Birkhäuser, Basel (1980)
4. Deneubourg, J., Aron, S., Goss, S., Pasteels, J.: The self-organizing exploratory pattern of the Argentine ant. Insect Behavior 3, 159–168 (1990)
5. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press, Cambridge (2004)
6. Miramontes, O., Solé, R.V., Goodwin, B.C.: Neural networks as sources of chaotic motor activity in ants and how complexity develops at the social scale. International Journal of Bifurcation and Chaos 11(6), 1655–1664 (2001)

A Model Based Ant Colony Design for the Protein Engineering Problem

Matteo Borrotti[1,2], Davide De Lucrezia[2], Giovanni Minervini[2], and Irene Poli[2,3]

[1] Department of Statistics, University of Bologna, Bologna, Italy
matteo.borrotti@unibo.it
[2] European Centre for Living Technology, University of Venice, Venice, Italy
{gminervini,davide.delucrezia}@ecltech.org
[3] Department of Statistics, University of Venice, Venice, Italy
irenpoli@unive.it

Abstract. In many experimental setting, we are concerned with finding the optimal experimental design, *i.e.* the configuration of predictive variables corresponding to an optimal value of the response. However, the high dimensionality of the search space, the vast number of variables and the economical constrains limit the ability of classical techniques to reach the optimum of a function. In this paper, we investigate the combination of statistical modeling and optimization algorithms to better explore the combinatorial search space and increase the performance of classical approaches. To this end, we propose a Model based Ant Colony Design (MACD) based on statistical modelling and Ant Colony Optimization. We apply the novel technique to a simulative case study related to Synthetic Biology.

Keywords: Evolutionary Optimization, Design of Experiments, Predictive Model, Ant Colony Optmization, Synthetic Proteins.

1 Introduction

In recent years, discrete optimization of experiments concerning high dimensional systems has become a crucial topic in many experimental fields. Biological systems represent a typical example as they involve a highly hierarchical organization with a large number of variables. When planning an experiment, one difficulty is the complex structure of interactions that needs to be considered in order to achieve an optimal value of the response. We propose a novel methodology combining the strength of two methodologies: Design of Experiments (DoE)[5] and Ant Colony Optimization [4]. The former deals with statistical modeling and fitting the relationship between the response and design variables, while the second is an efficient optimization strategy suitable in high demensionality. The resulting procedure, called Model based Ant Colony Design (MACD), improves upon the limits of the individual techniques enabling us to deal with the huge experimental space of the possible solutions.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 352–359, 2010.

We propose a simulation setup for assessing the performance of our method. Our aim is to apply the procedure to the Protein Engineering and Design (PED) problem, which involves multi-dimensional experimental space. One important aim of PED is to find modified proteins with novel properties. In paricular we develop new synthetic proteins concerning a huge discrete sequence space. MACD is employed on a simulation set up reflecting the nature of the experimental space and it is tested against different optimization algorithms. Our numerical results show that the method is fast and involves limited experimental runs.

2 The Methodological Approach

Let A be a discrete set of experimental inputs, where the cardinality of A is $|A| = n$. Our goal is to choose an $m - uple$ ($m \ll n$) of experimental conditions from A (with possible duplicates). The selected $m - uple$ must optimize the underlying objective. The MACD is tested and compared with different optimization algorithms using difference istances that simulate the features of the real problem.

2.1 Model Based Ant Colony Design (MACD)

Ant Colony Optimization (ACO)[4] is one of the best bio-inspired algorithms for discrete optimization. Within the ACO metaheuristc framework one of the currently best performing version is the $\mathcal{MAX} - \mathcal{MIN}$ Ant System ($\mathcal{MM}$AS)[6]. We implement the $\mathcal{MM}$AS and hybrid versions of the algorithm based on two different local search techniques: Iterative Improvement Algorithm and Simulated Annealing (SA)[3]. Finally, we implement the MACD. The novelty of our procedure is boosting of the best hybrid version of the ACO algorithm by a predictive statistical model. As a consequence, the resulting algorithm searches the design space more exhaustively. The following steps summarize our procedure:

1. Randomly generate and evaluate an initial population (*e.g.* 100) of $m-uples$;
2. Estimate a predictive statistical model based on the population of the available $m-uples$. Here, we use a linear regression model with binary predictive variables, which is estimated from data by the least squares method. This model does not include interactions between variables;
3. Select a new set of 100 $m-uples$ by the solution construction process implemented in the $\mathcal{MM}$AS. For this purpose, we create a graph where each node represents a specific domain. A solution is a path with length 4 composed of 4 objects connected by arches;
4. Identify the best predicted $m - uple$ and use it to start a local search by the Simulated Annealing (400 iterations). Make a prediction of the response value using the fitted statistical model;
5. If the predictive response value of the new solution is larger than the one selected in Step 4, the new solution replaces the old one in the population obtained at Step 2;
6. The pheromone matrices are updated using the Iteration-Best Ant;

7. Repeat steps from 3 to 6 until the stop criterion is satisfied. In our case, we stop after 100 iterations. When the stop criterion is satisfied, the last set of $m-uples$ proposed by the approach is chosen as the new set of candidate solutions to be tested;
8. The new set of candidate solutions is evaluated and included in the set of the $m-uples$ that have been already evaluated;
9. Repeat the steps from 2 to 8 for a fixed number of experiment generations. In our specific case, such a number is set to be 10.

The procedure describe above improves upon existing methods in two main directions:

1. Thanks to the statistical model, we can simulate the problem and move in the search space as many times as we want, improving the solutions step by step;
2. The iterated refinement of the predictive model provides the optimization algorithm with predictive capability of the model resulting in an increased accuracy during the optimization process.

2.2 The Protein Engineering Problem

One of the aims of Synthetic Biology is designing novel biological components (*i.e.* proteins, metabolic and regulatory networks) for medical, industrial and environmental applications. Particularly, proteins are important biological elements as they are ubiquitous in nature and they are responsible for major part of the biological tasks (*i.e.* catalysis, capability to enhance chemical reactions within a cell). A protein P_i is a sequence of monomers, called amino-acids, joined together to form a complex string. Each protein may differ in length, amino acid composition and sequence and is characterized by a well defined three-dimensional structure which in turn defines its function.

Starting from non-natural random sequences, we have designed a library of 95 random subsequences, called domains, each composed of 50 amino acids. The amino acid frequency used to generate random domains reflects the composition of natural proteins. Random domains are combinatorially assembled to generate full-length random proteins of 200 amino acids (4 domains per protein) to be subsequently screened for catalytic function. Thus, each individual protein can be considered as a string composed of 4 domains chosen among the 95 polymers with replacement. Accordingly, all possible permutations (with possible repetitions) of 95 elements in 4 positions are 95^4, approx 8.1×10^7, which represents the number of theoretically different full-length random proteins to be screened. The experimental evaluation of such a huge number is beyond technical reach since expression, purification and assessment of a protein may take from a few days to weeks. The main challenge is to develop effective methodologies to identify the best domains in the proper positions (a selection and ordering optimization problem) to construct functional proteins.

In order to be experimentally tested, candidate proteins should respect the following biological restrictions: (i) the number of cysteine residues should be at

most 9 and different from 5 and 7, since proteins with these features are hard to express and purify. (ii) The percentage of coil should not be larger than 70% otherwise proteins will hardly fold into stable tertiary structure. We implement these constrains in Step 3 (search phase) of the method described in the previous section.

3 Simulation Setting

In this section, we test the performance of four algorithms on simulated experiments. Particularly, we consider two different experiment generating processes in order to asses the quality of candidate solutions and test their performance. The following four approaches are applied to 100 independent simulated experiments:

- $\mathcal{MAX}-\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS), with colony size of 100 ants and evaporation factor 0.80;
- $\mathcal{MAX}-\mathcal{MIN}$ Ant System with Iterative Improvement Local Search (Local-$\mathcal{MM}$AS), with colony size of 100 ants and evaporation factor 0.80;
- $\mathcal{MAX}-\mathcal{MIN}$ Ant System with Simulated Annealing (SA-$\mathcal{MM}$AS), with colony size of 100 ants and evaporation factor 0.80;
- Model Based Ant Colony Design (see description in section 2.1).

The evaporation factor is too strong, in the first three algorithms, because the algorithms must reach good solutions in a few iterations. A preliminary study about optimal parameters configuration showed that the chosen values are the most performing.

3.1 The Data Generating Models (DGM)

As a benchmark functions to generate experiment evaluations, we choose the two functions described belove.

Polynomial regression model (PRM). This structure is described by a polynomial regression model with 380 main effects (*i.e.* the effects of one of the i, with $i = 1, \ldots, 95$, domains in each j, with $j = 1, \ldots, 4$, different positions) and 18 interactions between pairs of variables and 12 interactions among triplets. The interactions between pairs of variables and triples are obtained considering the 3 best domains for each positions and combining them in pairs and triplets. This model represents a protein fitness landscape dominated by strong interactions (*i.e.* epistasis) which occurs when the effect of one mutation depends on the presence of another [2]. This kind of fitness landscapes is characterized by ruggedness and local optima, the resulting simulative Polynomial regression model being formalized as follows:

$$y = \sum_{i=1}^{95}\sum_{j=1}^{4}\beta_{ij}x_{ij} + \alpha_1 x_{2,1}x_{95,2} + \alpha_2 x_{2,1}x_{49,3} + \alpha_3 x_{2,1}x_{95,4} + \alpha_4 x_{95,2}x_{49,3}$$
$$+ \alpha_5 x_{95,2}x_{95,4} + \alpha_6 x_{49,3}x_{95,4} + \alpha_7 x_{1,1}x_{93,2} + \alpha_8 x_{1,1}x_{48,3} + \alpha_9 x_{2,1}x_{94,4}$$

$$+ \alpha_{10}x_{93,2}x_{48,3} + \alpha_{11}x_{93,2}x_{94,4} + \alpha_{12}x_{48,3}x_{94,4} + \alpha_{13}x_{3,1}x_{94,2} + \alpha_{14}x_{3,1}x_{50,3}$$
$$+ \alpha_{15}x_{3,1}x_{1,4} + \alpha_{16}x_{94,2}x_{50,3} + \alpha_{17}x_{94,2}x_{1,4} + \alpha_{18}x_{50,3}x_{1,4} + \delta_1 x_{2,1}x_{95,2}x_{49,3}$$
$$+ \delta_2 x_{2,1}x_{95,2}x_{95,4} + \delta_3 x_{2,1}x_{49,3}x_{95,4} + \delta_4 x_{95,2}x_{49,3}x_{95,4} + \delta_5 x_{1,1}x_{93,2}x_{48,3}$$
$$+ \delta_6 x_{1,1}x_{93,2}x_{94,4} + \delta_7 x_{93,2}x_{48,3}x_{94,4} + \delta_8 x_{1,1}x_{48,3}x_{94,4} + \delta_9 x_{3,1}x_{94,2}x_{50,3}$$
$$+ \delta_{10}x_{3,1}x_{94,2}x_{1,4} + \delta_{11}x_{94,2}x_{50,3}x_{1,4} + \delta_{12}x_{3,1}x_{50,3}x_{1,4} \quad (1)$$

where the coefficients β_{ij} (where $i = 1, \ldots, 95$ and $j = 1, \ldots, 4$) are:

β_{i1} equally spaces in $[30, \ldots, -30]$;
β_{i2} equally spaces in $[-20, \ldots, 20]$;
β_{i3} equal to a parabolic function $-10z^2 + z + 30$ with z in $[-10, \ldots, 10]$;
β_{i4} equal to a parabolic function $10z^2 + z - 30$ with z in $[-10, \ldots, 10]$;
α, δ both positive and negative coefficients of interactions among pairs and triplets of the three best x_i for each j.

Furthermore, since each x_{ij} represents a specific domain in a particular position then x_{ij} is equal to 1 if the domain is in the considered sequence otherwise it is 0. The optimal solution is $x = (1, 95, 49, 95)$ with the response value equal to 184.961.

Polynomial sparse regression model (PSRM). This second formal structure to generate data represents the situation where some elements for each position j, in the protein sequence, highly influence the response of the system and the others are close to 0. This model closely represents an experimental protein fitness landscape where the major part of the protein sequences do not posses any function (zero fitness) whereas rare functional proteins are tightly clustered together [1]. The resulting simulative model is:

$$y = \sum_{i=1}^{95}\sum_{j=1}^{4} \beta_{ij}x_{ij} + \alpha_1 x_{54,1}x_{7,2} + \alpha_2 x_{54,1}x_{63,3} + \alpha_3 x_{54,1}x_{16,4} + \alpha_4 x_{7,2}x_{63,3}$$
$$+ \alpha_5 x_{7,2}x_{16,4} + \alpha_6 x_{63,3}x_{16,4} + \alpha_7 x_{48,1}x_{17,2} + \alpha_8 x_{48,1}x_{91,3} + \alpha_9 x_{48,1}x_{76,4}$$
$$+ \alpha_{10}x_{17,2}x_{91,3} + \alpha_{11}x_{17,2}x_{76,4} + \alpha_{12}x_{91,3}x_{76,4} + \alpha_{13}x_{12,1}x_{20,2} + \alpha_{14}x_{12,1}x_{84,3}$$
$$+ \alpha_{15}x_{12,1}x_{47,4} + \alpha_{16}x_{20,2}x_{84,3} + \alpha_{17}x_{20,2}x_{47,4} + \alpha_{18}x_{84,3}x_{47,4} + \delta_1 x_{54,1}x_{7,2}x_{63,3}$$
$$+ \delta_2 x_{54,1}x_{7,2}x_{16,4} + \delta_3 x_{54,1}x_{63,3}x_{16,4} + \delta_4 x_{7,2}x_{63,3}x_{16,4} + \delta_5 x_{48,1}x_{17,2}x_{91,3}$$
$$+ \delta_6 x_{48,1}x_{17,2}x_{76,4} + \delta_7 x_{17,2}x_{91,3}x_{76,4} + \delta_8 x_{48,1}x_{91,3}x_{76,4} + \delta_9 x_{12,1}x_{20,2}x_{84,3}$$
$$+ \delta_{10}x_{12,1}x_{20,2}x_{47,4} + \delta_{11}x_{20,2}x_{84,3}x_{47,4} + \delta_{12}x_{12,1}x_{84,3}x_{47,4} \quad (2)$$

As before, x_{ij} is equal to 1 when the domain is in the considered sequence and in a specific position, otherwise it is 0. In this case the elements that influence the response are randomly selected with uniform probability and their coefficients are drawn from a normal distribution $N(35, 10)$. The optimal solution is $x = (54, 7, 63, 16)$ with the response value equal to 232.426.

3.2 Results

Preliminary studies demostrated that the Local-$\mathcal{MM}$AS could not reach better solutions than the other approaches considered. Thus, we decided to use the

MACD with the more performing version of the $\mathcal{MM}$AS, the SA-$\mathcal{MM}$AS. It is important to highlight that the function evaluations used by the local search are counted when the number of function evaluations are determined.

In Fig. 1, we show function evaluations and their response values. We report the first 1000 evaluations. After the 500 experiment evaluations, the MACD has a larger probability to find good solutions than the other two algorithms. In the

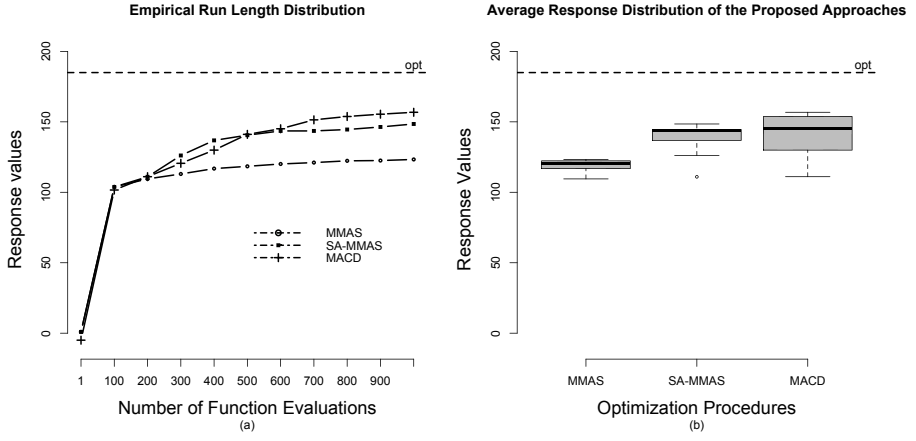


Fig. 1. Performance comparison between the MACD, the $\mathcal{MM}$AS and the SA-$\mathcal{MM}$AS. The candidate solutions are evaluated with the PRM. (a) shows the Empirical Run Length Distribution and (b) the Average Distribution through boxplot representation.

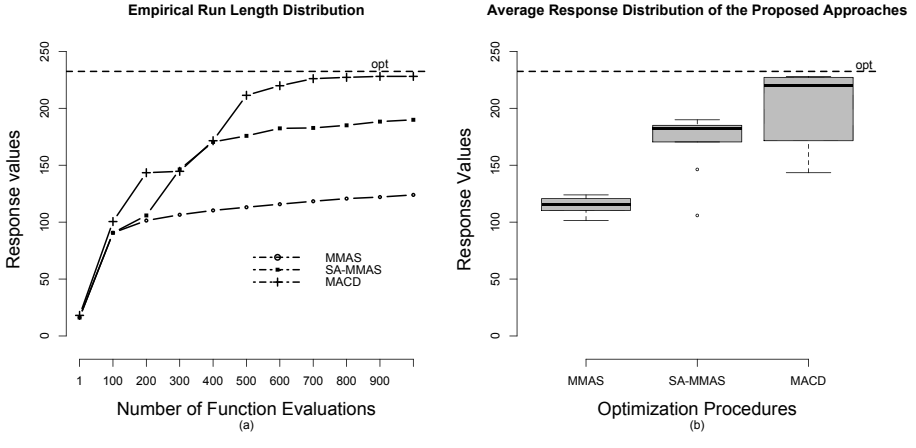


Fig. 2. Performance comparison between the MACD, the $\mathcal{MM}$AS and the SA-$\mathcal{MM}$AS. The candidate solutions are evaluated with the PSRM. (a) shows the Empirical Run Length Distribution and (b) the Average Distribution through boxplot representation.

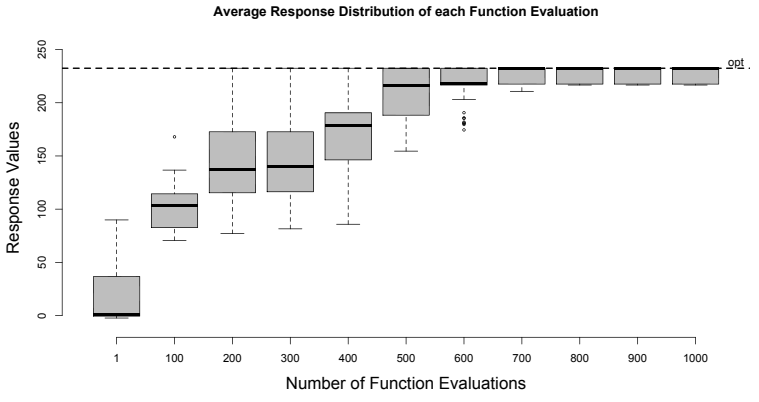


Fig. 3. Average Fitness Distribution of the MACD among the 1000 function evaluations. The candidate solutions are evaluated with the PSRM.

first function evaluations, our approach cannot reach good results due to the limited amount of data available at that point. However, after 400 experiment evaluations the predictive model starts to be more reliable. The performance of the MACD appears more evident for the PRSM, as one can see in Fig.2.

Fig.3 shows that the MACD is able to reach the optimum of the PSRM after only 200 function evaluations. The MACD outperforms the other approaches, which instead ignore the information provided by the predictive statistical model. In conclusion, the analysis of the performance of MACD shows that such a novel approach reaches good solutions using only a few iterations in both case studies.

4 Conclusion

The novel algorithm derived in this work can be effectively employed for optimizing an experimental design by a relative small number of experiment evaluations. In the simulation set-up considered here, the MACD reaches good solutions by only 1000 evaluations. Most notably, in the case of PSRM, the MACD achieves the optimal solution within 200 function evaluations.

The simulated landscapes have been designed to reflect concrete experimental problems encountered in the design and optimization of proteins, so that these results can be translated to an experimental setting with reasonable confidence. The two functions describe in Section 3.1 provide insights on the behaviour of the method in rugged experimental response surface with possible many local optima.

Within this framework, the proposed approach may be deployed to aid the search of novel proteins with a significant reduction of resources and time. Indeed, traditional methods routinely screen from 10.000 to 100.000 protein candidates before identifying robust leads. Instead, the application of our approach

may dramatically reduce the number of screening required to identify promising candidates. In addition, our algorithm provides an insight on the contribution of individual domains to the overall fitness.

In conclusion these results suggest that the integration of advanced statistical techniques with optimization algorithms may be profitably employed in optimization of experimental design with a large experimental space which is typical in biological experimental field.

Acknowledgements. Support from DICE (*Designing Informative Combinatorial Experiments*) project found by Fondazione di Venezia. We wish to thanks also Matteo Borrotti's PhD supervisor, Professor Alessandra Giovagnoli, University of Bologna. Furthermore we acknowledge ECLT (*European Centre for Living Technology*) group and we are grateful to Thomas Stützle and Mauro Birattari for providing useful comments on this work. We also wish to thanks the anonymous referees for the remarks and suggestions for improvements.

References

1. Aita, T., Iwakura, M., Husimi, Y.: A cross-section of the fitness landscape of dihydrofolate reductase. Protein Eng. 14(9), 633–638 (2001)
2. Bershtein, S., Segal, M., Bekerman, R., Tokuriki, N., Tawfik, D.S.: Robustness-epistasis link shapes the fitness landscape of a randomly drifting protein. Nature 444(7121), 929–932 (2006)
3. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35(3), 268–308 (2003)
4. Dorigo, M., Stützle, T.: Ant Colony Optimization. Bradford Book (2004)
5. Montgomery, D.C.: Design and Analysis of Experiments. John Wiley & Sons, Chichester (2006)
6. Stützle, T., Hoos, H.H.: $\mathcal{MAX} - \mathcal{MIN}$ ant system. Future Gener. Comput. Syst. 16(9), 889–914 (2000)

ACOPHY: A Simple and General Ant Colony Optimization Approach for Phylogenetic Tree Reconstruction

Huy Q. Dinh[1,2,⋆], Bui Quang Minh[1,⋆],
Hoang Xuan Huan[3], and Arndt von Haeseler[1]

[1] Center for Integrative Bioinformatics, Vienna, Max F. Perutz Laboratories, University of Vienna, Medical University of Vienna, University of Veterinary Medicine Vienna, Austria
[2] Gregor Mendel Institute of Molecular Plant Biology, Austrian Academy of Sciences, Vienna, Austria
[3] Faculty of Information Technology, College of Technology, Hanoi, Vietnam
huy.dinh@gmi.oeaw.ac.at, minh.bui@univie.ac.at, huanhx@vnu.edu.vn, arndt.von.haeseler@univie.ac.at

Abstract. We introduce ACOPHY, a novel framework to apply Ant Colony Optimization (ACO) for phylogenetic reconstruction. ACOPHY overcomes a main drawback of other attempts to reconstruct phylogenies by defining a compact ACO graph that is nicely coupled with the tree space. The proposed graph allows the ants to walk globally through the tree space. Thus, ACOPHY can be generally applied to all well-known optimality criteria in phylogenetics. We compared ACOPHY with the traditional phylogenetic method PHYLIP and obtained slightly better results. This is promising since our current implementation of ACOPHY is still at the proof of concept stage. We list a number of points where ACOPHY can be improved. Once the improvements are integrated, we hope for competitive performance against other recent phylogenetic inference methods.

Keywords: Phylogenetic reconstruction, ant colony optimization.

1 Introduction

Phylogenetic reconstruction is one of the main problems in biology [6]. In phylogenetic trees, contemporary organisms are located at the leaves of the tree, whereas the internal nodes represent extinct common ancestors. The topology of the tree depicts the evolutionary relationships between the organisms. The branch length reflects the amount of evolutionary divergence between the ancestor and its descendant. With the advent of molecular biology and DNA sequencing technologies, phylogenetic trees are nowadays often inferred from sequence data. The input data constitute a so-called multiple sequence alignment (MSA).

⋆ These authors contributed equally to the work.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 360–367, 2010.

It suffices to say that an MSA is a two-dimensional matrix of size $n \times m$ where n is the number of species and m is the alignment length. In MSA, columns are also called *sites*. All phylogenetic reconstruction approaches try to reconstruct a tree that explains the variation observed in the MSA.

For such data, phylogenetic reconstruction can be formulated as a combinatorial optimization problem. Phylogenetic methods are categorized according to the objective function they optimize: maximum parsimony (MP), maximum likelihood (ML), and minimum evolution (ME) [6]. The first two methodologies are character-based, that is to say trees are inferred directly from the MSA. Under MP, we want to find the tree(s) $\mathcal{T}$ that minimize the total tree length $L(\mathcal{T})$, the minimal number of character changes along $\mathcal{T}$ required to explain the observed differences in the alignment. For a given tree $\mathcal{T}$, $L(\mathcal{T})$ can be efficiently computed by the Fitch's algorithm [8]. The third category is called distance-based, i.e., trees are constructed from pairwise distances derived from the sequences of the MSA. For all objective criteria, finding the best trees was proven NP-complete [9]. This is mainly due to the exponential growth of the number of tree topologies with regards to the number of leaves [4].

Therefore, a variety of heuristics have been introduced to address this computationally difficult problem. The simple greedy algorithm is perhaps among the most widely used. These techniques have been implemented in a number of phylogenetic programs including PHYLIP, PHYML [7,10]. Greedy algorithms, however, are easily trapped into local optima (chapter 4, [6]). Therefore, global optimization heuristics have been proposed. These include iterated local search (IQPNNI [20]), simulated annealing (RAxML-SA [18]), genetic algorithm (MetaPIGA [13]), and tabu search (LEAPHY [21]).

Ant colony optimization (ACO [5]) is a recent heuristic framework inspired by the behaviors of real ant colonies. Recently, ACO has been applied exclusively for distance-based methods [1,15,3]. However, distance-based methods have the disadvantage of losing sequence information and requiring a reliable estimate of the distances. To the best of our knowledge there exist no character-based methods using ACO.

Here, we introduce ACOPHY, a novel ACO framework for phylogenetic reconstruction to overcome the drawback of previous works. ACOPHY can be used for character-based and distance-based objective functions. To this end, we define a compact graph structure allowing the ants to walk globally through the tree space. In the following, we present the ACOPHY and a proof of concept implementation of ACOPHY using MP. We perform a comparison of ACOPHY with the widely-used PHYLIP method using simulated data. Finally, we discuss future extensions of the proposed method.

2 ACOPHY

As defining a suitable graph structure is the most challenging task for applying ACO, we first propose the ACO graph that is nicely coupled with the tree space. We then follow the ACO tradition to define heuristic information, solution construction, pheromone update, and finally the overall ACOPHY algorithm.

2.1 Construction Graph

We define the ACO graph $\mathcal{G}$ (Fig. 1) consisting of $(n-2)^2$ nodes, where n is the number of species. Nodes correspond to points on the plane that are arranged in $n-2$ layers. The first layer has the single point with the coordinate of $(1,0)$. We call it the nest of the ants. The second layer contains three points $(2,1)$, $(2,0)$, and $(2,-1)$. In general, layer i contains all points (i,a_i), where a_i are integer numbers and $|a_i| < i$. Hence, the last layer contains $2n-5$ points. The nodes on the last layer are called food sources. The ants always walk on $\mathcal{G}$ from the nest to the food sources through consecutive layers. Hence, $\mathcal{G}$ is a weighted directed acyclic graph including arcs (or edges) that go from all nodes in layer i to all other nodes in layer $i+1$ with $i = 1, 2, \ldots, n-3$. The weights on the arcs correspond to the pheromone trails.

The defined graph $\mathcal{G}$ is related to the phylogenetic reconstruction as follows. We denote the set of n species as $X = \{*, 0, 1, \ldots, n-2\}$. The path, on which an ant travels from the nest to the food sources, contains $n-2$ nodes $(1,a_1), \ldots, (n-2, a_{n-2})$. This path corresponds to an integer vector $(a_1, a_2, \ldots, a_{n-2})$, where $a_i < |i|$ for $i = 1, \ldots, n-2$. This vector can be uniquely translated into an unrooted bifurcating tree containing the n species by the so-called Bandelt-Dress code [2] (see Fig. 1 for an example). Thus we have established an one-to-one correspondence

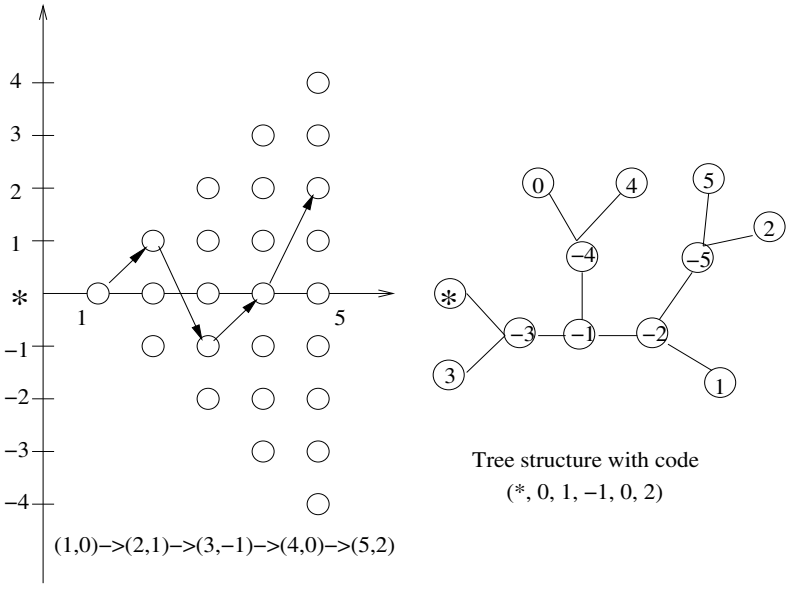


Fig. 1. The ACOPHY graph (on the left-hand side) for $n = 7$. The graph contains 5^2 nodes arranged inside a symmetric triangle. The arrows indicate the traveling path of an ant from the nest point $(1,0)$ to one of the food sources. The y-coordinates of the ant visiting nodes form the 5-element vector $(*, 0, 1, -1, 0, 2)$. This vector can be decoded into the 7-species tree depicted on the right-hand side.

between the ant traveling paths and the trees. Algorithms for translating the integer vectors into trees and vice versa were given in [2]. We refrain from detailing the algorithms but note that it works by stepwise addition. Firstly, a single initial sub-tree of 3 leaves $\{*, 0, 1\}$ is drawn where $*$ denotes the virtual root. Subsequently at each step i ($2 \leq i \leq n-2$), the species i will be added into the current sub-tree at the edge from the node a_i towards to the root. This is repeated until species $n-2$ is inserted.

Finally, it is easy to see that the number of paths from the nest to the food sources is $t(n) = 1 \cdot 3 \cdot \ldots \cdot (2n-5)$. This is exactly the number of unrooted bifurcating trees for n species [4]. Therefore, our graph $\mathcal{G}$ allows the ants to walk globally through the tree space within a compact structure of just $O(n^2)$ nodes and $O(n^3)$ arcs[1].

2.2 Pheromone Trail and Heuristic Information

For each arc $(i, a_i) \rightarrow (i+1, a_{i+1})$ we associate a pheromone trail $\tau[(i, a_i) \rightarrow (i+1, a_{i+1})]$. The heuristic information is defined in a more complex way as follows. Assuming that the ant is currently staying on layer i and the path, that this ant has traveled so far, is coded as the vector $(a_1, a_2, \ldots, a_i)$. It now continues to layer $i+1$ by finding an integer value a_{i+1}, where $|a_{i+1}| < i+1$. On the other hand, the vector $(a_1, a_2, \ldots, a_{i+1})$ corresponds to a sub-tree containing species $\{*, 0, 1, \ldots, i+1\}$. Under MP, one can efficiently compute this sub-tree length given $i+1$ sequences by the Fitch's algorithm [8]. We denote it by $L(a_1, a_2, \ldots, a_{i+1})$. The heuristic information can be intuitively defined as $1/L(a_1, a_2, \ldots, a_{i+1})$.

2.3 Solution Construction and Pheromone Update

For solution construction the ant at layer i with the current path $(a_1, a_2, \ldots, a_i)$ will select $(i+1, a_{i+1})$ as the next node with the probability

$$p(a_{i+1}|a_1, \ldots, a_i) = \frac{\tau[(i, a_i) \rightarrow (i+1, a_{i+1})]^\alpha / L(a_1, \ldots, a_{i+1})^\beta}{\sum_{|x|<i+1} \tau[(i, a_i) \rightarrow (i+1, x)]^\alpha / L(a_1, \ldots, a_i, x)^\beta}, \quad (1)$$

where α and β are two parameters determining the relative influence of pheromone trail and heuristic information, respectively.

For pheromone update we apply the MAX-MIN ant system [19]. Firstly, all pheromone trails evaporate with a factor of $\rho \in [0, 1]$:

$$\tau[arcs] \leftarrow \rho\tau[arcs], \forall arcs \in \mathcal{G}. \quad (2)$$

Then, we identify the best ant that found the best tree among the n_{ants} running ants. We now deposit an amount of pheromone along the path $(a_1^*, \ldots, a_{n-2}^*)$ of the best ant by

$$\tau[arcs] \leftarrow \tau[arcs] + (1-\rho)/L(a_1^*, \ldots, a_{n-2}^*), \forall arcs \in (a_1^*, \ldots, a_{n-2}^*). \quad (3)$$

[1] The exact number of arcs is $(n-3)[\frac{2}{3}(n-2)(2n-5)-1]$

2.4 The Overall ACOPHY Algorithm

Taking together we proceed as follows.

1. Initialize parameters and pheromone trails on all arcs to τ_{max}.
2. Initialize the best tree reconstructed by the greedy stepwise addition [6].
3. Iterate the following steps t_{ACO} times:
   (a) Let n_{ants} ants simultaneously walk on the graph from the nest to the food sources with the decision rule in section 2.3.
   (b) Update the pheromone trails following as described in section 2.3.
   (c) Apply the nearest neighbor interchange (NNI) [6] to improve the current best ant every, say, 100 iterations.
   (d) Update the global best tree if the current best ant shows a better score.

This procedure follows the ACO scheme except that we apply the tree rearrangement by NNI in step 2(c). Since NNI is a type of local search inside the tree space, NNI can be regarded as an extended local search with respect to the ACOPHY graph. NNI is simple but was shown very effective in practice [10].

3 Performance Study

3.1 Data Simulation

As a proof of concept, we implemented ACOPHY under the MP principle (source codes are available upon request). The ACOPHY parameters are shown in Table 1. To study its performance we compared ACOPHY with a classical parsimony method in the PHYLIP package, the *dnapars* program [7]. We could not obtain other ACO implementations for phylogenetics [1,15,3].

The accuracy of both methods is assessed by simulated data following a standard scheme in phylogenetics [10,20]. We generated 100 random true trees with 30 taxa (small datasets) and 100 taxa (large datasets) under the Yule-Harding distribution [11]. The branch lengths were drawn from an exponential distribution with mean values of 0.1. We then used Seq-Gen [16] to evolve sequences

Table 1. The parameter setup for ACOPHY

Parameter	Value	Description
t_{ACO}	10,000	number of ACOPHY iterations
n_{ants}	25	number of ants used in ACOPHY
τ_{min}	$1/2l$[a]	lower bound of pheromone trail
τ_{max}	$1/l$	upper bound of pheromone trail
α	1	relative influence of pheromone trail
β	2	relative influence of heuristic information
ρ	0.9	pheromone evaporation factor

[a] l is the score of the parsimony tree constructed by the greedy stepwise addition algorithm.

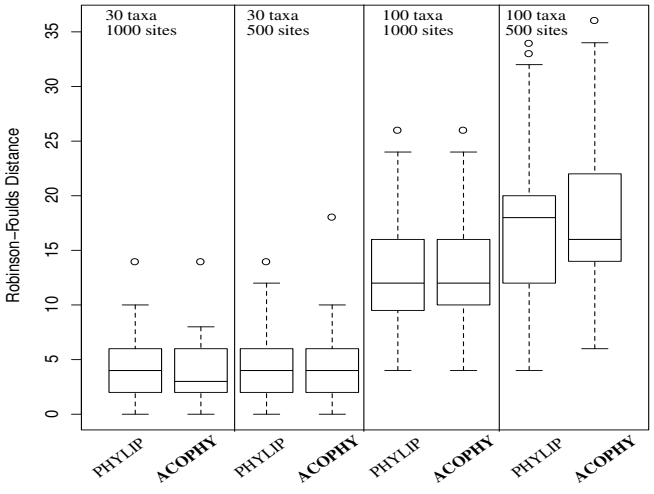


Fig. 2. Box-plots of the 100 Robinson-Foulds distances between the trees inferred by PHYLIP / ACOPHY and the simulated true trees for four test instances

along the trees according to the Juke-Cantor model [12]. We repeated this procedure for short alignments (500 sites) and long alignments (1000 sites).

We then ran ACOPHY 10 times on each alignment to account for the stochastic behavior of ACOPHY. The tree with the best score found in the 10 runs is considered as the final ACOPHY tree. For each alignment, we ran PHYLIP only once since it implemented a deterministic algorithm with the option of fixed input order of sequences.

We assessed the accuracy of ACOPHY and PHYLIP by two quantities. The first quantity is the number of times that the methods recovered the original true trees. The second is the Robinson-Foulds (RF) distance [17] between the inferred trees and the true trees, i.e., the number of branches that appear in one tree but not the other. Hence the smaller the RF distance was, the better the corresponding method performed.

3.2 Results

For (small, short)-datasets, PHYLIP and ACOPHY returned 17 and 19 correct trees, respectively. For (small, long)-datasets, these numbers are 19 and 22, respectively. Hence, ACOPHY slightly outperformed PHYLIP. For large datasets, both PHYLIP and ACOPHY were not able to recover any true trees.

Therefore, we computed the RF distances and depicted the RF distribution by box-plots (Fig. 2). From Fig. 2 it is clear that both methods performed comparable though ACOPHY showed a better median in two instances (small long and large short alignments). The increase in RF distances for large datasets compared to small datasets is mainly due to difficulties in reconstructing large

trees [20]. The MP scores found by both methods are significantly correlated with the Pearson correlation coefficient ~ 1. In particular for large long MSA ACOPHY reconstructed 13 trees having the MP score of $0.01 - 0.42\%$ smaller than those of PHYLIP. However, for 7 cases PHYLIP achieved better MP scores of $0.006 - 0.01\%$.

Next, we evaluate the consistency of ACOPHY using the 10 runs for each dataset. We observed that for the small datasets, all 10 runs returned the same tree. However, for 42 large long and 67 large short datasets ACOPHY found trees with highly different scores in the 10 runs. Hence, ACOPHY had difficulties in convergence that might be attributed to the insufficient number of iterations t_{ACO} or not enough phylogenetic information.

The running time of ACOPHY for large long alignments was roughly 2 hours per run on average, whereas PHYLIP needed only 10 minutes. This is expected because ACO-based methods are much slower than greedy algorithms. It is also because our code was currently not optimized.

4 Discussions

We have introduced the novel ACOPHY method using the ACO meta-heuristics for phylogenetic reconstruction. The core component of ACOPHY is the compact ACO graph. However, it is still qualified as a general graph that allows us to apply all optimality criteria. Despite the proof of concept implementation, ACOPHY was shown to perform as good as the well-known phylogenetic method PHYLIP.

ACOPHY can be further improved. Firstly, one can test different ACOPHY parameters instead of the default setting (Table 1). Secondly, we note that the current implementation of the Fitch's algorithm and the NNI search are inefficient. A better NNI strategy [10] can accelerate the method substantially, thus allowing us to apply NNI more frequently. Thirdly, the MAX-MIN ant system allows to re-initialize the pheromone trails if no better tree is found after a given number of iterations. Such feature can be easily integrated into ACOPHY to avoid 10 independent runs. More generally, one can incorporate a probabilistic stopping rule [20] to determine the required number of further iterations on-the-fly.

As mentioned, ACOPHY can be extended to ML and ME criteria. For ML, the computation becomes much more expensive. Therefore, parallel computing can be applied to speed up the computation [14]. Moreover, ACOPHY is being developed under a general-purpose phylogenetic library allowing to easily "plug-in" new heuristics. This framework will help us to integrate the aforementioned improvements and achieve a competitive performance with other recent phylogenetic methods.

Acknowledgments. B.Q.M., H.Q.D. and A.v.H. thank the Wiener Wissenschafts-, Forschungs- und Technologiefonds for financial support. H.Q.D. also thanks the support from Gregor Mendel Intitute, Vienna. We thank Anne Kupczok for critical reading of the manuscript.

References

1. Ando, S., Iba, H.: Ant algorithm for construction of evolutionary tree. In: Evolutionary Computation, vol. 2, pp. 1552–1557. IEEE Press, Los Alamitos (2002)
2. Bandelt, H.J., Dress, A.: Reconstructing the shape of a tree from observed dissimilarity data. Adv. Appl. Math. 7, 309–343 (1986)
3. Catanzaro, D., Pesenti, R., Milinkovitch, M.: An ant colony optimization algorithm for phylogenetic estimation under the minimum evolution principle. BMC Evol. Biol. 7 (2007)
4. Cavalli-Sforza, L.L., Edwards, A.W.F.: Phylogenetic analysis: Models and estimation procedures. Amer. J. Human. Genet. 19, 233–257 (1967)
5. Dorigo, M., Stuetzle, T.: Ant Colony Optimization. The MIT Press, Cambridge (2004)
6. Felsenstein, J.: Infering Phylogenies. Sinauer Associates, Sunderland (2004)
7. Felsenstein, J.: PHYLIP – Phylogeny Inference Package (version 3.2). Cladistics 5, 164–166 (1989)
8. Fitch, W.M.: Toward defining the course of evolution: Minimum change for a specific tree topology. Syst. Zool. 20, 406–416 (1971)
9. Foulds, L.R., Graham, R.L.: The Steiner problem in phylogeny is NP-complete. Adv. Appl. Math. 3, 43–49 (1982)
10. Guindon, S., Gascuel, O.: A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst. Biol. 52, 696–704 (2003)
11. Harding, E.F.: The probabilities of rooted tree-shapes generated by random bifurcation. Adv. Appl. Prob. 3, 44–77 (1971)
12. Jukes, T.H., Cantor, C.R.: Evolution of protein molecules. In: Munro, H.N. (ed.) Mammalian Protein Metabolism, vol. 3, pp. 21–123. Academic Press, New York (1969)
13. Lemmon, A.R., Milinkovitch, M.C.: The metapopulation genetic algorithm: An efficient solution for the problem of large phylogeny estimation. Proc. Natl. Acad. Sci. USA 99, 10516–10521 (2002)
14. Minh, B.Q., Vinh, L.S., von Haeseler, A., Schmidt, H.A.: pIQPNNI: Parallel reconstruction of large maximum likelihood phylogenies. Bioinformatics 21, 3794–3796 (2005)
15. Perretto, M., Lopes, H.S.: Reconstruction of phylogenetic trees using the ant colony optimization paradigm. Genet. Mol. Res. 4, 581–589 (2005)
16. Rambaut, A., Grassly, N.C.: Seq-Gen: An application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. Comput. Appl. Biosci. 13, 235–238 (1997)
17. Robinson, D.F., Foulds, L.R.: Comparison of phylogenetic trees. Math. Biosci. 53, 131–147 (1981)
18. Stamatakis, A.P.: An efficient program for phylogenetic inference using simulated annealing. In: Online Proceedings of the 4th IEEE International Workshop on High Performance Computational Biology, Denver (2005)
19. Stuetzle, T., Hoos, H.: Max-min ant system. Future Gener. Comp. Sy. 16, 889–914 (2000)
20. Vinh, L.S., von Haeseler, A.: IQPNNI: Moving fast through tree space and stopping in time. Mol. Biol. Evol. 21, 1565–1571 (2004)
21. Whelan, S.: New approaches to phylogenetic tree search and their application to large numbers of protein alignments. Syst. Biol. 56, 727–740 (2007)

ACS Searching for D_{4t}-Hadamard Matrices

Víctor Álvarez, José Andrés Armario, María Dolores Frau, Félix Gudiel, Belén Güemes, Elena Martín, and Amparo Osuna

University of Sevilla, Sevilla, Spain
{valvarez,armario,mdfrau,gudiel,bguemes,emartin,aosuna}@us.es

Abstract. An Ant Colony System (ACS) looking for cocyclic Hadamard matrices over dihedral groups D_{4t} is described. The underlying weighted graph consists of the rooted trees described in [1], whose vertices are certain subsets of coboundaries. A branch of these trees defines a D_{4t}-Hadamard matrix if and only if two conditions hold: (i) $I_i = i - 1$ and, (ii) $c_i = t$, for every $2 \leq i \leq t$, where I_i and c_i denote the number of i-paths and i-intersections (see [3] for details) related to the coboundaries defining the branch. The pheromone and heuristic values of our ACS are defined in such a way that condition (i) is always satisfied, and condition (ii) is closely to be satisfied.

Keywords: Cocyclic Hadamard matrix, ant colony system, i-path, i-intersection.

1 Introduction

Hadamard matrices are square matrices with entries ± 1 such that their rows are pairwise orthogonal. It is easy to prove that the size of Hadamard matrices must be 1, 2 or a multiple of 4. Nevertheless, it is an open question whether Hadamard matrices exist for every size $4t$. This is known as the *Hadamard Conjecture*. Recommended references on Hadamard matrices and their applications are [10] and more recently [12].

Actually, constructing Hadamard matrices is a difficult problem of optimization. That being so, different heuristics have been proposed to look for Hadamard matrices (see [2,6,5] for instance), but they all seem to run in exponential time $O(2^t)$.

One of the most promising techniques for solving the Hadamard Conjecture is the cocyclic approach [12], since both the search space and the time required for testing the Hadamard character of a matrix are significantly improved in this framework [12,1]. Among others, dihedral groups D_{4t} seem to provide a large amount of cocyclic Hadamard matrices (see [12] or [4], for instance). This is the reason for which we will focus on this family of groups. In the sequel, for short, cocyclic matrices over D_{4t} will be simply denoted as D_{4t}-matrices.

Experimental results in [1] suggest that one might restrict to look for D_{4t}-Hadamard matrices satisfying the *central distribution*. (this notion will be explained in detail in Section 2).

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 368–375, 2010.

Our aim here is to use the ideas of ant colony optimization (in the sequel, ACO for brevity) in order to design an ant colony system looking for D_{4t}-Hadamard matrices satisfying the central distribution.

We organize the paper as follows. Notations and previous results are introduced in Section 2. Section 3 is devoted to the description of our ACS. Last section is devoted to examples and conclusions.

2 Describing the Rooted Trees

In what follows, we will adopt the notations and results introduced in [1], which we describe now.

Consider the dihedral group D_{4t}, given by the presentation

$$< a, b | a^{2t} = b^2 = (ab)^2 = 1 >$$

and ordering $g_1 = 1 = a^0 < \ldots < g_{2t} = a^{2t-1} < g_{2t+1} = b < \ldots < g_{4t} = a^{2t-1}b$.

A 2-cocycle over D_{4t} consists in a map $f : D_{4t} \times D_{4t} \to \{1, -1\}$ such that

$$f(g_i, g_j) f(g_i g_j, g_k) = f(g_j, g_k) f(g_i, g_j g_k), \qquad \forall\, g_i, g_j, g_k \in D_{4t}.$$

A cocyclic matrix over D_{4t} (in the sequel, D_{4t}-matrix) consists in a matrix $M_f = (f(g_i, g_j))$, f being a 2-cocycle over D_{4t}.

A basis for 2-cocycles over D_{4t} is given by $\mathcal{B} = \{\partial_a, \ldots, \partial_{a^{2t-3}b}, \beta_1, \beta_2, \gamma\}$, where ∂_g denotes the elementary coboundary related to the element g, that is

$$\partial_g(g_i, g_j) = \delta_g(g_i)\delta_g(g_j)\delta_g(g_i g_j) \qquad \text{for} \qquad \delta_g(g_i) = \begin{cases} -1, & g = g_i \\ 1, & g \neq g_i \end{cases}$$

$\beta_1(a^i b^l, a^j b^k) = (-1)^{ij}$, $\beta_2(a^i b^l, a^j b^k) = (-1)^{lk}$ and

$$\gamma(a^i b^l, a^j b^k) = \begin{cases} -1, l = 0 \text{ and } i + j \geq 2 \\ -1, l = 1 \text{ and } i < j \\ 1, \text{otherwise} \end{cases}$$

We will consider only D_{4t}-matrices of the type $M_f = M_{\partial_{i_1}} \ldots M_{\partial_{i_w}} \cdot R$, in terms of some coboundary matrices $M_{\partial_{i_j}}$ and the matrix $R = M_{\beta_2} M_\gamma$. There is computational evidence that most of D_{4t}-Hadamard matrices are of this type (see [9,3] for instance).

Furthermore, the cocyclic Hadamard test (which asserts that a cocyclic matrix is Hadamard if and only if the summation of each row but the first is zero, [13]) runs four times faster for this type of D_{4t}-matrices, since it suffices to check whether the summation of rows from 2 to t are zero. For clarity in the exposition, from now on, the rows whose summation is zero are simply termed *Hadamard rows*.

In [3] the Hadamard character of a cocyclic matrix is described in an equivalent way, in terms of *generalized coboundary matrices*, *i-walks* and *intersections*. We reproduce now these notions.

The *generalized coboundary matrix* $\bar{M}_{\partial_j}$ related to a elementary coboundary ∂_j consists in negating the j^{th}-row of the matrix M_{∂_j}. Note that negating a row of a matrix does not change its Hadamard character. As it is pointed out in [3], every generalized coboundary matrix $\bar{M}_{\partial_j}$ contains exactly two negative entries in each row $s \neq 1$, which are located at positions (s, i) and (s, e), for $g_e = g_s^{-1} g_i$. We will work with generalized coboundary matrices from now on.

A set $\{\bar{M}_{\partial_{i_j}} : 1 \leq j \leq w\}$ of generalized coboundary matrices defines an *i-walk* if these matrices may be ordered in a sequence $(\bar{M}_{l_1}, \ldots, \bar{M}_{l_w})$ so that consecutive matrices share exactly one negative entry at the i^{th}-row. Such a walk is called an *i-path* if the initial and final matrices do not share a common -1, and an *i-cycle* otherwise. As it is pointed out in [3], every set of generalized coboundary matrices may be uniquely partitioned into disjoint maximal i-walks. It is clear that every maximal i-path contributes two negative occurrences at the i^{th}-row.

An *i-intersection* is a position in which R and $\bar{M}_{\partial_{i_1}} \ldots \bar{M}_{\partial_{i_w}}$ share a common -1 in the i^{th}-row.

From the definitions above, a characterization of Hadamard rows (consequently, of Hadamard matrices) may be easily described in terms of the number c_i of i-paths and the number I_i of i-intersections.

Proposition 1. *[3] The i^{th} row of a D_{4t}-matrix $M = M_{\partial_{i_1}} \ldots M_{\partial_{i_w}} \cdot M_{\beta_2} \cdot M_{\gamma}$ is Hadamard if and only if*

$$c_i - I_i = t - i + 1, \quad 2 \leq i \leq t. \tag{1}$$

It should be desirable to know the way in which coboundaries combine to form i-paths and i-intersections. These questions have already been answered.

On one hand, as it is described in [3], for $2 \leq i \leq t$, a maximal i-walk consists of a maximal subset in $(M_{\partial_1}, \ldots, M_{\partial_{2t}})$ or $(M_{\partial_{2t+1}}, \ldots, M_{\partial_{4t}})$ formed from matrices $(\ldots, M_j, M_k, \ldots)$ such that $j \pm (i-1) \equiv k \bmod 2t$.

On the other hand, in [[1], Lemma 3, p.208], a complete distribution of the coboundaries in $\mathcal{B}$ which produce an intersection at a given row is described in a table, so that coboundaries which produce the same negative occurrence at a row are displayed vertically in the same column. For clarity in the reading, we note the generalized coboundary $\bar{M}_{\partial_i}$ simply by i:

row	*coboundaries*
2	$2t$ $2t+1$
3	2t-1 $2t$ 2 $2t+1$ 2t+2
$4 \leq k \leq t$	2t-k+2 $2t-k+3$ $\ldots$ $2t-1$ 2 $\ldots$ $k-2$ $2t$ k-1 $2t+1$ 4t-k+2 $2t+2$ $\ldots$ $2t+k-3$ $4t-k+1$ $\ldots$ $4t-2$ $2t+k-2$ 2t+k-1

Notice that the boxed coboundary matrices do not produce any intersection at the precedent rows. Furthermore, $\bar{M}_{\partial_t}$, $\bar{M}_{\partial_{t+1}}$, $\bar{M}_{\partial_{3t}}$ and $\bar{M}_{\partial_{3t+1}}$ do not produce any intersection at all.

Though formally there are many distributions $(c_2, I_2), \ldots, (c_t, I_t)$ satisfying the Hadamard test (1), in [1] there is computational evidence that the distribution $(c_i, I_i) = (t, i-1)$ provides a large density of D_{4t}-Hadamard matrices, for $2 \leq t \leq 5$. This is termed the *central distribution*, and is expected to provide many D_{4t}-Hadamard matrices for larger values of t as well. The tables in [1] support this idea.

As it was already described in [1], the search space in the central distribution $(c_i, I_i) = (t, i-1)$, $2 \leq i \leq t$, may be represented as a forest of two rooted trees of depth $t-1$. Each level of the tree is identified to the correspondent row of the cocyclic matrix at which intersections are being counted, so that the roots of the trees are located at level 2 (corresponding to the intersections created at the second row of the cocyclic matrix).

This way the level i contains those coboundaries which must be added to the father configuration in order to get the desired $i-1$ intersections at the i^{th}-row, for $2 \leq i \leq t$.

The root of the first tree is ∂_{2t}, whereas the root of the second tree is ∂_{2t+1}, since these are the only coboundaries which may give an intersection at the second row.

As soon as one of these coboundaries is used, the other one is forbidden, since otherwise a second intersection would be introduced at the second row.

Now one must add some coboundaries to get two intersections at the third row. Notice that one and just one of $\{\partial_{2t}, \partial_{2t+1}\}$ is already used, whereas the other is forbidden.

Successively, in order to construct the nodes at level k, one must add some of the correspondent boxed coboundaries of the table, since the remaining coboundaries are either used or forbidden.

Some pictures representing these forests for $2 \leq t \leq 4$ are included in [1].

Summing up, if we assume that we are looking for D_{4t}-Hadamard matrices satisfying the central distribution, two conditions must hold:

$$I_i = i - 1 \tag{2}$$

$$c_i = t \tag{3}$$

Condition (2) means that one should find a branch in the trees above reaching the bottom level, t. But just a few of these branches define a D_{4t}-Hadamard matrix. Attending to condition (3), among all branches reaching the level t, one should select only those which inherits a subset of coboundaries such that eventually combined with some of the matrices $\bar{M}_{\partial_t}$, $\bar{M}_{\partial_{t+1}}$, $\bar{M}_{\partial_{3t}}$ and $\bar{M}_{\partial_{3t+1}}$ do produce exactly t i-walks, $2 \leq i \leq t$.

3 Defining the ACS

Sometimes solving an optimization problem is so difficult, that one restricts oneself to obtain not necessarily a global optimum but just a local one. Heuristic procedures are concerned with this purpose. A special kind of heuristics are

those emulating natural behaviours, such as evolutionary algorithms (inspired by biological evolution, see [11]) and ant colony optimization (simulating the pheromone model of ants, see [8]).

Although some heuristics (in terms of image restoration [6] or even evolutionary computation [2,5]) have already been used for constructing Hadamard matrices, as far as we know ant colony optimization has not been used for this purpose yet. This is our concern here. We are going to define an ant colony system (in the sequel, ACS [7]).

To this end, we need to define a weighted graph G modeling the problem, as well as the values τ_{ij}, η_{ij}, ϕ, ρ, τ_0, q_0, α, β and the function Δ_{ij} proper of ACSs.

The underlying graph G modeling our optimization problem consists of the rooted trees T_1 and T_2 described in the Section 2.

A D_{4t}-matrix M_f satisfies the central distribution if and only if the conditions (2) and (3) are satisfied.

As it was noted before, condition (2) means that the coboundaries generating f define a branch in one of our rooted trees, ending at the bottom level, t.

Unfortunately, there is no such geometric translation for condition (3).

For instance, the rooted trees associated to the case $t = 5$ consists of 84 branches each, all of them ending at the bottom level, 5. This means that the subsets of coboundaries associated to each of these branches satisfy the condition (2). Unfortunately, only 14 of these 84 branches give raise to D_{4t}-Hadamard matrices, since only 14 of the corresponding subsets of coboundaries (eventually combined with some of $\{\partial_5, \partial_6, \partial_{15}, \partial_{16}\}$) satisfy the condition (3). These branches are listed in the table below (the subsets of added ∂_i are listed in brackets).

Branches from T_1	Branches from T_2
(10)(9)(3, 13, 18)(14)[16]	(11)(2)(3)(4, 14, 17)[6, 15]
(10)(9)(3, 13, 18)(17)[15]	(11)(2)(13)(7)[6, 16]
(10)(9)(3, 13, 18)(4, 7, 14)[5, 6, 16]	(11)(2)(13)(4, 14, 17)[5, 6]
(10)(9)(3, 13, 18)(4, 7, 17)[5, 6, 15]	(11)(2)(3, 13, 18)(17)[5]
(10)(12)(8)(17)[5, 16]	(11)(2)(3, 13, 18)(17)[6, 15, 16]
(10)(12)(8)(4, 7, 14)[15, 16]	(11)(2)(8, 13, 18)(7)[]
(10)(12)(13)(4, 7, 14)[6, 16]	(11)(9)(8)(7, 14, 17)[5, 15]
(10)(12)(3, 8, 13)(4)[15]	(11)(9)(13)(4)[5, 16]
(10)(12)(3, 8, 13)(4)[5, 6, 16]	(11)(9)(13)(4, 7, 17)[5, 6]
(10)(12)(3, 8, 18)(17)[]	(11)(9)(3, 13, 18)(4)[]
(10)(2, 9, 12)(8)(14)[5, 6, 15, 16]	(11)(9)(8, 13, 18)(17)[6]
(10)(2, 9, 12)(8)(4, 7, 17)[5, 16]	(11)(9)(8, 13, 18)(17)[5, 15, 16]
(10)(2, 9, 12)(13)(4)[16]	(11)(12)(3, 8, 18)(4)[6]
(10)(2, 9, 12)(13)(4)[5, 6, 15]	(11)(12)(3, 8, 18)(7)[5]
(10)(2, 9, 12)(18)(4, 7, 17)[5, 6]	(11)(12)(3, 8, 18)(4, 14, 17)[6, 15, 16]
(10)(2, 9, 12)(3, 8, 13)(14)[6, 16]	(11)(12)(3, 8, 18)(7, 14, 17)[5, 15, 16]

Condition (3) is not satisfied by the remainder branches. This means that given such a branch, for every possible subset of $\{\partial_5, \partial_6, \partial_{15}, \partial_{16}\}$, there exists a row i such that $c_i \neq t = 5$. For instance, the tables below show the values c_i when the set of coboundaries defining the branch $(10)(9)(3,8,13)(17)$ are combined with the subset indicated.

Added ∂_i	c_2	c_3	c_4	c_5
	4	5	5	4
[5]	5	5	5	4
[6]	5	5	4	4
[15]	5	4	6	5
[16]	4	6	5	5

Added ∂_i	c_2	c_3	c_4	c_5
[5, 6]	5	5	4	4
[5, 15]	6	4	6	5
[5, 16]	5	6	5	5
[6, 15]	6	4	5	5
[6, 16]	5	6	4	5
[15, 16]	4	5	6	6

Added ∂_i	c_2	c_3	c_4	c_5
[5, 6, 15]	6	4	5	5
[5, 6, 16]	5	6	4	5
[5, 15, 16]	5	5	6	6
[6, 15, 16]	5	5	5	6
[5, 6, 15, 16]	5	5	5	6

Now we are in conditions to define the values τ_{ij} and η_{ij}.

The heuristic value η_{ij} is defined attending to condition (3), so that $\eta_{ij} = \dfrac{1}{\sum_{k=2}^{l} |t - c_k|}$, where l indicates the level of vertex v_j. This way, the nearer the path is to the central distribution $(t, \ldots, t)$, the higher η_{ij} is.

Initially, $\tau_{ij} = \tau_0 = 0.25$. We define $\Delta_{ij} = \tau_{ij} + 3\frac{\phi}{\rho}|\tau_{ij} - \tau_0|$, so that the pheromone values are updated for the set of edges belonging to the largest path among the ants' traversals by means of the formula $\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij} = \tau_{ij} + 3\phi|\tau_{ij} - \tau_0|$. Thus, although the local pheromone update has taken place before, the final value of τ_{ij} is greater or equal than its initial value, excepting the value of the last edge of the path, which is settled to 0. Consequently, this edge will never be used again. This is not a source of difficulties for the algorithm, since our graph consists of trees, and hence there is no edge going further.

We set $\alpha = 0.75$ and $\beta = 0.25$, so that the relative importance of pheromone versus heuristic information is settled accordingly to our purposes, since we need to get a branch reaching the bottom level t.

The evaporation rate is $\rho = 0.5$, the pheromone decay coefficient is $\phi = 0.5$, the probability of choosing the "best" edge is $q_0 = 0.75$. These values have been fixed experimentally.

Every iteration, the ants looks for good paths. For paths reaching at least level $t-1$, a local search is performed. This procedure consists in an exhaustive search from the vertex of the branch located at level $t-1$, combining with the 16 subsets of $\{\partial_t, \partial_{t+1}, \partial_{3t}, \partial_{3t+1}\}$. No matter the result of the search is, the weight of the edge arriving to the level $t-1$ is settled to 0, since now there is no chance to get not traversed paths from it. Whenever a D_{4t}-Hadamard matrix is found while performing the local search, it is added to a list *hadmat*.

The algorithm stops as soon as a the list *hadmat* is not empty.

We include now a pseudo-code for our ACS.

Algorithm 1. *ACS searching for D_{4t}-Hadamard matrices.*

```
Input: an integer t
Output: a list hadmat of D_{4t}-Hadamard matrices.

hadmat ← ∅
while hadmat is empty {
    for i from 1 to m do {
       trav_i ← traversal of ant i
       if trav_i reaches level t − 1 then local_search(trav_i)
       set to 0 the weight of the last edge of trav_i
    }
    actualize the weight of the edges of the best traversal
}
hadmat
```

In the following section we include some executions and final comments.

4 Examples

All the calculations of this section have been worked out in MATHEMATICA 4.0, running on a *Pentium IV 2.400 Mhz DIMM DDR266 512 MB*.

We have fixed $m = 5$, so that every iteration 5 traversals are performed.

For every $3 \leq t \leq 8$, we have performed 10 trials looking for D_{4t}-Hadamard matrices. The table below shows the fewest, largest and average number of iterations required, the best, worst and average time required and the minimum, maximum and average number of D_{4t}-Hadamard matrices found in these calculations.

t	Fewest	Largest	Av.Iter.	Best	Worst	Av.Time	min	max	Av.
3	1	1	1	$0.02''$	$0.11''$	$0.08''$	10	10	10
4	1	1	1	$0.48''$	$0.66''$	$0.59''$	14	22	19.6
5	1	1	1	$1.56''$	$2.34''$	$1.86''$	2	4	3
6	1	1	1	$1.01''$	$7.43''$	$3.59''$	1	4	2.5
7	2	14	6.1	$21.013''$	$3'18''$	$1'20''$	1	2	1.1
8	1	2	1.2	$1.93''$	$36.31''$	$15.41''$	1	2	1.1

For values $t \geq 9$, the algorithm does not find a D_{4t}-Hadamard matrix in reasonable time (more than 100 iterations are required). Nevertheless, the traversals reach the bottom level t most of times. Unfortunately, the subsets of coboundaries do not produce the required number of i-paths, for some row i. The table below shows how many of 500 traversals for $t = 9$ have ended at the corresponding level, as well as the amount of traversals for which the difference $\sum_{i=2}^{t} |t - c_i|$ is the indicated.

Level			$\sum_{i=2}^{t} \lvert t - c_i \rvert$												
6	7	8	14	12	11	10	9	8	7	6	5	4	3	2	1
10	15	475	1	5	9	14	39	106	122	61	52	66	17	7	1

Consequently, most of the traversals (475 from 500) satisfy the condition (2). This means that the definition of the pheromone values τ_{ij} is fine.

On the other hand, the condition (3) is never satisfied. This fact suggests that the formula for the heuristic values η_{ij} should be redefined somehow. We will deal with this problem in a near future.

Acknowledgments. All authors are partially supported by the research projects FQM–296 and P07-FQM-02980 from Junta de Andalucía and MTM2008-06578 from Ministerio de Ciencia e Innovación (Spain).

References

1. Álvarez, V., Armario, J.A., Frau, M.D., Gudiel, F., Osuna, A.: Rooted trees searching for cocyclic Hadamard matrices over D_{4t}. In: Bras-Amorós, M., Høholdt, T. (eds.) AAECC-18. LNCS, vol. 5527, pp. 204–214. Springer, Heidelberg (2009)
2. Álvarez, V., Armario, J.A., Frau, M.D., Real, P.: A genetic algorithm for cocyclic Hadamard matrices. In: Fossorier, M.P.C., Imai, H., Lin, S., Poli, A. (eds.) AAECC 16. LNCS, vol. 3857, pp. 144–153. Springer, Heidelberg (2006)
3. Álvarez, V., Armario, J.A., Frau, M.D., Real, P.: A system of equations for describing cocyclic Hadamard matrices. J. of Comb. Des. 16(4), 276–290 (2008)
4. Álvarez, V., Armario, J.A., Frau, M.D., Real, P.: The homological reduction method for computing cocyclic Hadamard matrices. J. Symb. Comput. 44, 558–570 (2009)
5. Álvarez, V., Frau, M.D., Osuna, A.: A genetic algorithm with guided reproduction for constructing cocyclic Hadamard matrices. In: ICANNGA 2009. LNCS, vol. 5495, pp. 150–160. Springer, Heidelberg (2009)
6. Baliga, A., Chua, J.: Self-dual codes using image resoration techniques. In: Bozta, S., Sphparlinski, I.E. (eds.) AAECC 14. LNCS, vol. 2227, pp. 46–56. Springer, Heidelberg (2001)
7. Dorigo, M., Gambardella, L.M.: Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Trans. on Evol. Comp. 1(1), 53–66 (1997)
8. Dorigo, M., Stützle, T.: Ant colony optimization. The MIT Press, Cambridge (2004)
9. Flannery, D.L.: Cocyclic Hadamard matrices and Hadamard groups are equivalent. J. Algebra 192, 749–779 (1997)
10. Hedayat, A., Wallis, W.D.: Hadamard Matrices and Their Applications. Ann. Stat. 6, 1184–1238 (1978)
11. Holland, J.H.: Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor (1975)
12. Horadam, K.J.: Hadamard matrices and their applications. Princeton University Press, Princeton (2006)
13. Horadam, K.J., de Launey, W.: Generation of cocyclic Hadamard matrices. In: Computational algebra and number theory (Sydney, 1992). Math. Appl, vol. 325, pp. 279–290. Kluwer Acad. Publ., Dordrecht (1995)

Ant Based Semi-supervised Classification

Anindya Halder[1], Susmita Ghosh[2], and Ashish Ghosh[1]

[1] Center for Soft Computing Research, Indian Statistical Institute, Kolkata, India
{anindya_t,ash}@isical.ac.in
[2] Dept. of Computer Science & Engg., Jadavpur University, Kolkata, India
susmitaghoshju@gmail.com

Abstract. Semi-supervised classification methods make use of the large amounts of relatively inexpensive available unlabeled data along with the small amount of labeled data to improve the accuracy of the classification. This article presents a novel 'self-training' based semi-supervised classification algorithm using the property of aggregation pheromone found in natural behavior of real ants. The proposed algorithm is evaluated with real life benchmark data sets in terms of classification accuracy. Also the method is compared with two conventional supervised classification methods and two recent semi-supervised classification techniques. Experimental results show the potentiality of the proposed algorithm.

1 Introduction

Traditional machine learning methods for pattern classification require sufficient number of labeled data to assign an unlabeled pattern to a certain class according to its characteristics. However labeled patterns are often difficult, costly, and/or time consuming to obtain, as they require the efforts of experienced human annotators. Whereas unlabeled data relatively easy to get. Semi-supervised learning (classification) [5] methods make use of the large amount of available unlabeled data, along with the labeled data, to improve the classification accuracy. As semi-supervised classification requires less human intervention and produces better results, it is of great interest to the machine learning researchers in recent years.

A variety of semi-supervised learning methods are developed; they can be broadly categorized as follows, self-training [17,20], co-training [4], transductive support vector machines [1,6,7], graph-based methods [2,3], and Expectation Maximization with generative mixture models [15] etc.

In this article, a novel semi-supervised algorithm is proposed based on aggregation pheromone density which is inspired by the natural behavior of real ants and other social insects. The social insects' behaviors such as finding the best food source, building of optimal nest structure, brooding, protecting the larva, guarding etc. show intelligent behavior on the swarm level. *Ant Colony Optimization* (ACO) [8] and *Aggregation Pheromone Systems* (APS) [19] are computational algorithms modeled on the behavior of ant colonies. ACO [8] algorithms are designed to emulate ants' behavior of laying pheromone on the

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 376–383, 2010.

ground while moving to solve optimization problems. Pheromone is a type of chemical emitted by an organism to communicate between members of the same species. Pheromone, which is responsible for clumping or clustering behavior in a species and brings individuals into closer proximity, is termed as aggregation pheromone. Thus, aggregation pheromone causes individuals to aggregate around good positions which in turn produces more pheromone to attract individuals of the same species. In APS [19], a variant of ACO, this behavior of ants is used to solve real parameter optimization problems.

Though a large number of techniques exists for ant based unsupervised classification (i.e clustering) in the literature [11], only few attempts (*antminer* and it's different variants) [13] have been made for (supervised) classification. But no ant colony based semi-supervised classification is present in the literature as to the best of our knowledge.

As mentioned earlier *Aggregation Pheromone Systems* [19] is used for continuous function optimization where aggregation pheromone density is defined by a function in the search space. Inspired by the aggregation pheromone system found in ants and other similar agents, in earlier work, attempts are made for solving clustering [9] and classification [10] with encouraging results. Motivated form the earlier research, in this article a novel ant based semi-supervised classification algorithm using aggregation pheromone has been proposed.

The rest of the paper is organized as follows. Section 2 describes the detail descriptions of the proposed ant based semi-supervised classification method using aggregation pheromone system. Details of the experiments and analysis of results are provided in Section 3, and finally conclusions are drawn in Section 4.

2 Proposed Methodology: Aggregation Pheromone Density Based Semi-Supervised Classification(APSSC)

Consider a data set D with m classes and at least some small number of labelled data patterns from each class which, by our assumption, forms m homogeneous groups/colonies of ants in the *training/lebelled set* L. Also there are (relatively large) $|U|$ number of unlabeled data patterns in the *unlabeled set* U. Let $\boldsymbol{x}_1^u$, $\boldsymbol{x}_2^u$, $\boldsymbol{x}_3^u$, ..., $\boldsymbol{x}_{|U|}^u$ be the unlabeled data points represented as *unlabeled ants* a_1^u, a_2^u, a_3^u, ..., $a_{|U|}^u$ respectively.

Consider, $\boldsymbol{x}_1^{l_i}$, $\boldsymbol{x}_2^{l_i}$, $\boldsymbol{x}_3^{l_i}$, ..., $\boldsymbol{x}_{|C_i^O|}^{l_i}$, as the given *training data* patterns in the i^{th} *original training class* C_i^O. These patterns are considered as a population of $|C_i^O|$ number of ants represented as $a_1^{l_i}$, $a_2^{l_i}$, $a_3^{l_i}$, ..., $a_{|C_i^O|}^{l_i}$ respectively. Hence, an ant $a_j^{l_i}$ represents the j^{th} training data pattern $\boldsymbol{x}_j^{l_i} \in C_i^O$.

Initially at iteration $t = 0$ when no unlabeled pattern is added (latter demonstrated) to any of the *original training class*, then i^{th} *total training class/colony* C_i^T is the same as the i^{th} *original training class/colony* C_i^O.

Each labeled pattern ant emits pheromone at its neighborhood. The intensity of pheromone emitted by the k^{th} individual labeled ant $a_k^{l_j} \in C_j^T$ located at $\boldsymbol{x}_k^{l_j}$ decreases with its distance from $\boldsymbol{x}_k^{l_j}$. Thus the pheromone intensity at a point

closer to $\boldsymbol{x}_k^{l_j}$ is more than those at other points that are farther from it. To achieve this, the pheromone intensity emitted by ant $a_k^{l_j} \in C_j^T$ is modeled by a Gaussian distribution. Hence effect of the emitted pheromone density on the i^{th} unlabeled ant a_i^u (located at $\boldsymbol{x}_i^u$) at iteration $(t+1)$ due to k^{th} ant $a_k^{l_j} \in C_j^T$ located at $\boldsymbol{x}_k^{l_j}$ is given by the following equation:

$$\Delta\tau^{t+1}(\boldsymbol{x}_k^{l_j}, \boldsymbol{x}_i^u) = \exp^{-\frac{d(\boldsymbol{x}_k^{l_j}, \boldsymbol{x}_i^u)^2}{2\delta^2}} \tag{1}$$

where, δ denotes the spread of Gaussian function and $d(\boldsymbol{x}_j, \boldsymbol{x})$ is the Euclidean distance between $\boldsymbol{x}_k^{l_j}$ and $\boldsymbol{x}_i^u$.

The average effect of the emitted (aggregated) pheromone density on the i^{th} unlabeled ant a_i^u due to j^{th} *total training colony* C_j^T at iteration $(t+1)$ is given by

$$\Delta\overline{\tau}_{ij}^{t+1} = \frac{1}{|C_j^T|} \sum_{\boldsymbol{x}_k^{l_j} \in C_j^T} \Delta\tau^{t+1}(\boldsymbol{x}_k^{l_j}, \boldsymbol{x}_i^u); \quad \forall \ \ i, j. \tag{2}$$

Then pheromone density τ_{ij}^{t+1} due to j^{th} colony C_j^T on the i^{th} unlabeled (pattern) ant at iteration $(t+1)$ is updated according to the following equation.

$$\tau_{ij}^{t+1} = \rho\tau_{ij}^t + \Delta\overline{\tau}_{ij}^{t+1} \quad \forall \ \ i, j \tag{3}$$

where ρ is the evaporation constant.

After pheromone density is updated the membership of each unlabeled ant a_i^u for approching/belonging to each colony (class) C_j^T is computed as follows.

$$\mu_{ij}^{t+1} = \frac{\tau_{ij}^{t+1}}{\sum_{k=1}^{m} \tau_{ik}^{t+1}} \quad \forall \ \ i, j. \tag{4}$$

Once the membership of all the unlabeled ants are determined, they are *evaluated* to be temporarily added to the taring set for the next iteration. The evaluation is done as follows. If the maximum membership value $\max_j(\mu_{ij}^{t+1})$ (among all class/colony) of any unlabeled ant a_i^u is greater than a predefined threshold value MT then that unlabeled ant a_i^u is temporarily added (for the next iteration) to the taring set k (colony) for which membership is highest. This is represented by the following equations.

$$if \quad \max_j(\mu_{ij}^{t+1}) > MT \ \ then$$

$$C_k^T = C_k^O \cup \boldsymbol{x}_i^u; \quad where, \ k = \arg\max_j(\mu_{ij}^{t+1}) \tag{5}$$

Note that addition of the ant to the colony k is done temporarily for the next iteration, and in subsequent iterations depending on the current membership values it will be added to the appropriate colony or may not be included in any

colony. Hence in each iteration (re)assignment of the unlabeled ants occur and the algorithm stops when there is no (re)assignment. This is done by computing the colony centers and if the corresponding colony centers in two successive iterations are equal then we can say that there is no (re)partition. At that time we can say that colony formation by the unlabeled ants is over and the unlabeled ants are stabilized. This means either they have joined any colony with sufficient confidence (greater than MT), or (rest) have not joined any colony (with sufficient confidence). The unlabeled ants which have joined in any colony are now considered as the training points, and thus the size of the training set increases with the help of unlabeled points.

After the colony formation (by the unlabeled pattern), the patterns are tested as follows. If the test pattern ant a_t at $\boldsymbol{x_t}$ appears in the system, the average aggregation pheromone density (at the location of that new ant a_t) by the colony C_i^T is given by (as in equation 2)

$$\Delta\overline{\tau}_{ti} = \frac{1}{|C_i^T|} \sum_{x_j \in C_i^T} \exp^{-\frac{d(\boldsymbol{x}_j, \boldsymbol{x_t})^2}{2\delta^2}}. \tag{6}$$

The test ant a_t will move towards a colony for which the average aggregation pheromone density (at the location of the test ant) is higher than that of other colonies. Hence finally the said ant will join the colony governed by the following equation.

$$ColonyLabel(\boldsymbol{x_t}) = \arg\max_i(\Delta\overline{\tau}_{ti}). \tag{7}$$

Thus each of the test ant will join a colony and the corresponding label of the colony will be the class label of that test pattern (ant). The proposed semi-supervised aggregation pheromone density based classification (APSSC) algorithm is given below (Algorithm 1).

3 Experimental Evaluation

3.1 Data Sets

For the purpose of our study, we used a number of artificially generated data,and real life data sets. Among them only five typical real life data sets are reported in this article. Four among them are from the UCI repository[14], and Telugu Vowel data is from [16]. To test the classification accuracy, five percent of data is taken out randomly from a data set to form the initial training set and the rest is considered as the unlabeled set. The process is repeated 10 times. The reported results are obtained considering the unlabeled data as the test set. A summery about the data sets is given in Table 1.

3.2 Methods Compared

The proposed method is compared with two traditional classifier multi layer perceptron (MLP) [12], and Support Vector Machine (SVM) [18] and also with

Algorithm 1. Aggregation Pheromone density based Semi-Supervised Classification (APSSC)

***begin_self_training*()**
Initialize: Iteration counter $t \leftarrow 0$; $\tau_{ij}^0 \leftarrow 0,\ \forall\ i, j$.
repeat
  for each unlabeled ant a_i^u located at $\boldsymbol{x_i^u}$ **do**
    for each *total training colony* C_j^T **do**
      Calculate the average aggregation pheromone density $\Delta\overline{\tau}_{ij}^{t+1}$ on the i^{th} unlabeled ant a_i^u due to all ants in *total training colony* C_j^T at iteration $(t+1)$ using equation (2).
      Update pheromone density τ_{ij}^{t+1} due to j^{th} colony C_j^T on the i^{th} unlabeled (pattern) ant at interaction $(t+1)$ by equation (3).
    end for
    for each *total training colony* C_j^T **do**
      Compute the membership μ_{ij}^{t+1} of each unlabeled ant a_i^u for approching/belonging to the each colony (class) C_j^T at interaction $(t+1)$ using equation (4).
    end for
    if $\max_j(\mu_{ij}^{t+1}) > MT$ **then**
      Add the unlabeled ant a_i^u to the appropriate training colony using equation (5).
    else
      Do not add the unlabeled ant a_i^u to any training colony.
    end if
  end for
  $t \leftarrow t+1$.
until $<$ *StoppingCriteria* $>$
end_self_training
***begin_testing*()**
for each test ant a_t located at $\boldsymbol{x_t}$ **do**
  for each colony C_i^T **do**
    Calculate the average aggregation pheromone density at location $\boldsymbol{x_t}$ due to all ants in colony C_i^T using equation (6).
  end for
  Compute the $ColonyLabel(\boldsymbol{x_t})$ of the ant a_t by equation (7). // Ties are broken arbitrarily.
end for
end_testing

Table 1. Summery of the data sets used for the experiments

Data set	Classes	Dimensions	Pattern	Labeled pattern
Telugu vowel	6	3	871	5%
Balance Scale	3	4	625	5%
Sonar	2	60	208	5%
WBC	2	9	683	5%
Ionosphere	2	34	351	5%

two semi-supervised techniques, namely semi-supervised classification by low density separation (LDS) [6], and concave-convex procedure for transductive support vector machine (CCCP-TSVM) [7]. Note that compared methods have number of parameters. We have experimantally adjusted the parameters such the classification accuracy is optimum.

3.3 Role of the Parameters

The proposed method has three parameters namely δ, ρ , and MT.

Here δ is the spread of the Gaussian. For the optimal performance of the proposed method, we have experimented with wide range of δ for each data set. The δ value, for which the best result in terms of classification accuracy occurs, is reported in Table 2 and that selected δ value is put in Table 3. Note that for a wide range of δ, values of the performance measure is observed to be fixed at nearly constant value or varies a little.

To see the effect of the evaporation coefficient ρ, we varied the value of the ρ and experimentally fix the value of the ρ. With larger values of ρ, system uses information of the pheromone density of the past cycles more than with the smaller values of ρ. In general performance is found to be almost constant or varies a little in the range [0.7-0.98].

MT indicates degree of confidence of an unlabeled pattern (for belonging to a certain class) we want to allow to temporarily add the unlabeled pattern in the training set. More is the value of the MT, more confidence unlabeled pattern will be added to the training set, but no of pattern added to the taring set will be less. We have varied the value of the MT in the range [0.4, 0.99] and experimentally set the value for optimum performance of the classifiers. In general large value (> 0.7) is observed to work fine.

The selected parameters are shown in Table 3 for each data set.

Table 2. Summery of the experimental results in terms of percentage accuracy

Methods	Telugu Vowel	Balance Scale	Sonar	WBC	Ionosphere
MLP	70.49638	81.3131	59.54546	92.78891	79.69972
SVM	65.71342	76.54858	55.5556	95.146379	77.5736
LDS	73.7933292	85.5219	62.8509299	96.8146048	91.2455193
CCCP-TSVM	78.80682	86.4926	65.735	97.8146048	93.524
APSSC	81.74	87.987239	67.66708	97.07289	89.24026

3.4 Experimental Results and Analysis

The average results for 10 simulation runs of all the algorithms are reported in Table 2. As mentioned before, the proposed APSSC algorithm has three parameters, and the experimentally determined values of the parameters are put in Table 3.

Table 3. Summery of the selected parameter values of APSSC method

Methods	Telugu Vowel	Balance Scale	Sonar	WBC	Ionosphere
δ	11.8	0.3	0.43	3.2	0.04
ρ	0.7	0.7	0.76	0.7	0.7
MT	0.8	0.85	0.9	0.85	0.95

For real life data sets, proposed APSSC observed to perform better in terms of classification accuracy in three cases namely Telugu Vowel, Balance Scale and Sonar data. For all other cases the performances of APSSC is quite close to that of the best one.

Note that all the semi-supervised classifiers clearly dominate the supervised classifiers (MLP and SVM). Though the percentage of training sample in both the cases (supervised and semi-supervised) is the same, the use of the unlabeled patterns really helps to gain the accuracy in semi supervised case.

Execution time is the least for CCCP-TSVM for most of the data sets. However, the execution time of the proposed algorithm is moderate.

4 Conclusions

This article presents a novel semi-supervised classification algorithm based on the property of the aggregation pheromone found in natural behavior of real ants. The performances of the proposed method is compared with two supervised (MLP and SVM) and two semi-supervised classification (LDS and CCCP-TSVM) techniques. Experiments were carried out with different kinds data sets. Experimental results justify the potentiality of the proposed APSSC algorithm in terms of classification accuracy with moderate execution time.

Future work of the proposed method may be directed towards solving real world problem for classification and also to adaptive determination of the parameters.

Acknowledgements. Support of the Department of Science and Technology, Govt. of India to the Center for Soft Computing Research (CSCR) through its IRHPA scheme is thankfully acknowledged by Mr. Anindya Halder, Research Scholar, CSCR, Indian Statistical Institute, Kolkata, India.

References

1. Bennett, K.P., Demiriz, A.: Semi-supervised support vector machines. In: Advances in Neural Information Processing Systems, pp. 368–374. MIT Press, Cambridge (1998)
2. Blum, A., Chawla, S.: Learning from labeled and unlabeled data using graph mincuts. In: Proc. 18th Intl. Conf. on Machine Learning, pp. 19–26. Morgan Kaufmann, San Francisco (2001)

3. Blum, A., Lafferty, J., Rwebangira, M.R., Reddy, R.: Semi-supervised learning using randomized mincuts. In: Proc. of the 21st Intl. Conf. on Machine Learning, pp. 97–104 (2004)
4. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on Computational learning theory, pp. 92–100. Morgan Kaufmann Publishers, San Francisco (1998)
5. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning, Adaptive Computation and Machine Learning. The MIT Press, Cambridge (2006)
6. Chapelle, O., Zien, A.: Semi-supervised classification by low density separation. In: Proc. of the 10^{th} Intl. Workshop on Artificial Intelligence and Statistics, pp. 57–64 (2005)
7. Collobert, R., Sinz, F., Weston, J., Bottou, L., Joachims, T.: Large scale transductive svms. Journal of Machine Learning Research 7, 1687–1712 (2006)
8. Dorigo, M., Stützle, T.: Ant Colony Optimization. Prentice Hall of India Private Limited, New Delhi (2005)
9. Ghosh, A., Halder, A., Kothari, M., Ghosh, S.: Aggregation pheromone density based data clustering. Information Sciences 178(13), 2816–2831 (2008)
10. Halder, A., Ghosh, A., Ghosh, S.: Aggregation pheromone density based pattern classification. Fundamenta Informaticae 92(4), 345–362 (2009)
11. Handl, J., Meyer, B.: Ant-based and swarm-based clustering. Swarm Intelligence 1, 95–113 (2007)
12. Haykin, S.: Neural Networks: A Comperehensive Foundation, 2nd edn. Prentice Hall, Englewood Cliffs (1999)
13. Liu, B., Abbass, H.A., McKay, B.: Density-based heuristic for rule discovery with ant-miner. In: Proc. of 6th Australasia-Japan Joint Workshop on Intelligence Evololution System, Canberra, Australia, pp. 180–184 (2002)
14. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. University of California, Department of Information and Computer Sciences, http://www.ics.uci.edu/~mlearn/MLRepository.html
15. Nigam, K., Mccallum, A.K., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning 39(2-3), 103–134 (2000)
16. Pal, S.K., Majumder, D.D.: Fuzzy sets and decision making approaches in vowel and speaker recognition. IEEE Transactions on Systems, Man, and Cybernetics 7, 625–629 (1977)
17. Rosenberg, C., Hebert, M., Schneiderman, H.: Semi-supervised self-training of object detection models. In: Seventh IEEE Workshop on Applications of Computer Vision, pp. 29–36 (2005)
18. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2002)
19. Tsutsui, S.: Ant colony optimization for continuous domains with aggregation pheromones metaphor. In: Proc. of the 5th Intl. Conf. on Recent Advances in Soft Computing, United Kingdom, pp. 207–212 (December 2004)
20. Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. In: Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics, pp. 189–196 (1995)

Automatic Generation of Optimised Working Time Models in Personnel Planning

Volker Nissen and Maik Günther

Technical University of Ilmenau, Chair of Information Systems in Services, Ilmenau, Germany
volker.nissen@tu-ilmenau.de, maik.guenther@gmx.de

Abstract. Retail is traditionally labour-intensive. Demand-oriented workforce management has great significance due to the amount of competition which enforces a strict cost management while keeping a good service level. Thus, highly flexible working time models are of particular importance. Our project addresses the question how to automatically and simultaneously assign staff to workstations and generate optimised working time models under constraints and on the basis of fluctuating personnel demand. The planning is completed for an entire year in order to assess adapted versions of the evolution strategy and particle swarm optimisation. A commercial constructive method is used as benchmark.

Keywords: integrated personnel planning, metaheuristics.

1 Introduction

The ability to adapt personnel assignment to changing requirements is of critical importance in workforce management (WFM). An interesting option are automatically generated flexible working time models that are based on demand while respecting certain constraints. The targeted benefits are cost reduction through improved utilisation of employee time, reduction of overtime and idle time, a rise in employee motivation and, thus, an increase of turnover through a higher level of service. The traditional approach to WFM in separate planning steps can be very inefficient. Therefore, an integrated design of working time models and staff schedules is suggested in this paper. More specifically, we adapt and compare the evolution strategy (ES) and particle swarm optimisation (PSO) for this integrated planning task. A commercial constructive approach is used as benchmark. While constructive approaches are generally popular in personnel scheduling and rostering [3], the metaheuristics performed well in related scheduling problems from logistics [7]. The research goals we pursue are twofold. First, we aim for good solutions to a meaningful practical application that is relevant in industries such as logistics, retail and call centers. Second, we want to contribute to the comparison of modern metaheuristics on problems of realistic complexity. Section 2 highlights the real-world case, before related work is presented in section 3. The subsequent sections introduce the solution methods which are then applied and compared in section 7.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 384–391, 2010.

2 Description of the Real-World Problem from a Retailer

This practical case concerns personnel planning in the department for ladies' wear at a department store. For current benchmarks and real data reference is made to [5]. We assume a set of employees $\mathcal{E} = \{1, \ldots, E\}$, a set of workstations $\mathcal{W} = \{1, \ldots, W\}$ and a discrete timeframe $\mathcal{T}$ with index $t = 0, \ldots, T-1$. Each period t of the range has a length l_t greater than zero.

$$l_t > 0 \quad \forall t \in \mathcal{T} \tag{1}$$

The assignment of an employee to a workstation is controlled using the binary variable x_{ewt}.

$$x_{ewt} = \begin{cases} 1 & \text{if employee } e \text{ is assigned to workstation } w \text{ at period } t \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The store is open Monday to Saturday from 10:00 to 20:00 and closed on Sunday and holidays. The availability of the employees is determined using the binary variable a_{et}.

$$a_{et} = \begin{cases} 1 & \text{if employee } e \text{ is available at period } t \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

There are 15 employees, assigned to two workstations (till and sales), with all employees trained for both stations. The personnel demand d_{wt} is given in one-hour intervals and is determined based on past data. A minimal and maximal number of employees per workstation and period is set. The demand d_{wt} of employees per workstation and period cannot be negative.

$$d_{wt} \geq 0 \qquad \forall w \in \mathcal{W} \text{ and } \forall t \in \mathcal{T} \tag{4}$$

There are many factors influencing planning, such as regulations, employee availability and time sheets. Because of fluctuations in demand, sub-daily workstation changes are allowed. As a hard constraint, working time models must begin and end on the hour. Also, sub-daily workstation changes are only possible on the hour. Moreover, an employee e can only be associated with a workstation w in the period t if he or she is actually present. Additionally, an employee can only be designated to one workstation at a time.

$$\sum_{w=1}^{W} x_{ewt} = a_{et} \qquad \forall e \in \mathcal{E} \text{ and } \forall t \in \mathcal{T} \tag{5}$$

There are also soft constraints. Their violation is penalised with error points that reflect the companies requirements as inquired through interviews. If a discrepancy arises from the workstation staffing target d_{wt}, error points P_d are generated for the duration and size of the erroneous assignment.

$$P_d = \sum_{t=0}^{T-1} \sum_{w=1}^{W} (c_{dn} + c_{do} + c_{du}) l_t \left| \left(\sum_{e=1}^{E} x_{ewt} \right) - d_{wt} \right|,$$

with: (6)

$c_{do} > 0$ if w is overstaffed at t and $d_{wt} > 0$, else $c_{do} = 0$

$c_{dn} > 0$ if w is overstaffed at t and $d_{wt} = 0$, else $c_{dn} = 0$

$c_{du} > 0$ if w is understaffed at t and $d_{wt} > 0$, else $c_{du} = 0$

Six employment contracts exist with a planned weekly working time between 10 and 40 hours. During weeks with bank holidays the planned working time s_e is reduced by a proportional factor h. The effective weekly working time i_e for an employee should not exceed the contractually agreed number of hours. Each minute in excess is punished with error points c_w.

$$P_w = c_w \sum_{week=1}^{52} \sum_{e=1}^{E} (i_e - s_e * h), \tag{7}$$

with $c_w = 0$ if $s_e * h - i_e \geq 0$, $c_w = 1$ else.

The automatically generated working time models should not be shorter than 3 hours or longer than 9 hours. Any violation leads to error points c_t per employee and day. The sum of these error points for the planning horizon is P_t. Working time models must not be split up during a working day, with violations leading to error points c_c per employee and day. The sum of these error points for the planning horizon is P_c. For an optimal coverage of personnel demand sub-daily workstation changes are required. However, to avoid an excessive number r_e of rotations for any employee c_r error points arise for such workstation changes.

$$P_r = c_r \sum_{e=1}^{E} r_e \tag{8}$$

Therefore, the objective function to be minimised becomes:

$$minP = P_d + P_w + P_t + P_c + P_r. \tag{9}$$

Historical data is available for a complete calendar year, so that an entire year (8.760 one hour time slots) can be planned ahead, resulting in a very complex search space of 131.400 decision variables. In practice, also shorter planning horizons (month) are employed. However, the full year plan helps the company to better understand on a more strategic level how well it can cope with demand using current staff.

A two-dimensional matrix of employees and time slots is applied to represent a solution for PSO and ES. The meaning of the matrix elements is as follows:

- 0: Store is closed or employee is absent (holiday, training, illness).
- 1: Employee is assigned to workstation 1.
- 2: Employee is assigned to workstation 2.
- 3: Employee is generally available but not dispatched in staffing

3 Related Work

Staff scheduling is a hard optimisation problem. Garey and Johnson [4] demonstrate that even simple versions of staff scheduling problems are NP-complete. Tien and Kamiyama [11] prove that practical staff scheduling is generally more complex than the TSP which is itself NP-hard. Sauer and Schumann [10] suggest a constructive approach specifically designed for retail. Unfortunately, it is not able to consider more than one workstation or sub-daily workstation rotations. Therefore, it cannot be applied to the practical case discussed here. Prüm [9] creates working time models parallel to assignment planning for problem instances in retail. Again, only one workstation is present and sub-daily job rotations are not included. The results indicate that problems of realistic size can not be successfully solved with exact methods. Therefore, in the following sections we focus on different heuristic approaches.

4 Constructive Method

As a benchmark, we use a commercial WFM software. (We cannot mention the name for legal reasons.) This tool delivers an adequate constructive method, capable of solving the problem. The software is in regular use at around 300 companies. All restrictions are supported and the error points associated with soft constraints can be entered into the software. Unfortunately, no code was available and no information was provided as to how the working time models are generated. Thus, it must be considered a black box. However, we were supported by the software manufacturer, so errors in software handling can be excluded.

5 Particle Swarm Optimisation

For our application, PSO [8] had to be adapted to the combinatorial domain. The idea to abandon velocity was taken from Chu, Chen and Ho [2]. Following suggestions in [6] on a similar scheduling problem, a gBest topology is applied, where each particle is a neighbour of every other particle. The following pseudocode presents an overview of our PSO.

01: Initialise the Swarm
02: Evaluate the Particles of the Swarm
03: Determine pBest for each Particle and gBest
04: Loop
05: For $i = 1$ to Number of Particles
06: Calculate new Position // 4 Actions for Calculation
07: Repair the Particle
08: Evaluate the Particle
09: If f(new Position) $\leq f$(pBest) then pBest = new Position // new pBest
10: If f(pBest) $\leq f$(gBest) then gBest = pBest // new gBest
11: Next i
12: Until Criterion

The swarm is initialized with valid solutions w.r.t. the hard constraints. pBest represents the best position found so far by the particle while gBest is the best position of all particles. In each iteration the new particle position is determined by traversing all dimensions and executing one of the following actions with predefined probability. The probability distribution was heuristically determined in prior tests.

- No change (p_1=9.98%): The workstation already assigned remains.
- Random workstation (p_2=0.02%): A workstation is (uniformly) randomly determined and assigned. Assignments respect employee availability.
- pBest workstation (p_3=30%): The corresponding workstation is assigned from pBest (individual component).
- gBest workstation (p_4=60%): The corresponding workstation is assigned from gBest (social component).

According to our tests, the behaviour of PSO is relatively insensitive to changes of p_1, p_3, and p_4, but very sensitive to p_2. The optimal value for p_2 depends on the problem size (smaller probabilities for larger problems). Too much randomness is destructive, but some is required to avoid premature convergence. Consequently, we varied p_2 in very small steps in pre-tests to arrive at the current suggestion. The characteristics of PSO have not been changed with these modifications. There are merely changes in the way to determine a new particle position, so that the calculation of velocity is not needed. PSO and ES both terminate after 400,000 inspected solutions to allow for a fair comparison.

Preliminary tests showed that repairing the solutions created by PSO could be beneficial. Our repair heuristic (not detailed here for reasons of space) corrects violations of soft constraints in the following order, based on the observed frequency of error occurrences: 1. Overstaffing 2. Understaffing 3. More than one working time model per day 4. Violation of minimum length of a working time model 5. Violation of maximum length of a working time model 6. Elemination of unnecessary workstation rotations.

6 Evolution Strategy

The ES was originally developed by Rechenberg and Schwefel [1] and soon applied to continuous parameter optimisation problems. Mutation is the main search operator employed in ES. Our application is of a combinatorial nature, though, which requires some adaptation of the ES. The pseudocode below presents an overview of the implemented ES.

The ES-population is initialized with valid solutions w.r.t the hard problem constraints. Ten alternative recombination variants were evaluated in a pre-test. The best performance was achieved with a rather simple form that is based on the classical one-point crossover. The same crossover point is determined at random for all employees (row) of a solution and the associated parts of the parents are exchanged to create an offspring.

01: Initialise the Population with μ Individuals
02: Repair the Population
03: Evaluate the μ Individuals
04: Loop
05: Recombination to generate λ Offspring
06: Mutate the λ Offspring
07: Repair the λ Offspring
08: Evaluate all repaired Individuals
09: Selection ((μ+λ) or (μ,λ))
10: Until Criterion

An offspring is mutated by picking an employee at random and changing the workstation assignment for a time interval chosen at random. It must be ensured, though, that valid assignments are made w.r.t. the hard problem constraints. The number of employees selected for mutation follows a $(0;\sigma)$-normal distribution. Results are rounded and converted to positive integer numbers. The mutation stepsize sigma is controlled self-adaptively using a log-normal distribution and intermediate recombination, following the standard scheme of ES [1].

After mutation, the same repair heuristic is applied to individuals to remove constraint violations as in PSO. (μ,λ)-selection (comma-selection) as well as (μ + λ)-selection (plus-selection) are used and different population sizes. The best solution found during an experimental run is always stored and updated. It represents the final solution of the run. Following suggestions in the literature [1], the ratio μ/λ is set to 1/5 - 1/7 during the experiments.

7 Results and Discussion

All heuristics were tested on the retailer problem with the objective of minimising error points under the given constraints. The implementation was done in C# on a 2.66 GHz quad core PC with 4 GB RAM. Table 1 presents the results. The runs using PSO and ES were repeated 30 times for each parameter set. One calculation (10 minutes) is sufficient for the constructive method, because the same result would be achieved each repetition. PSO and ES require roughly 6 hours for a single run with 400,000 fitness calculations. The constructive heuristic can be regarded as a benchmark, since it is actually used in many companies for staff planning. Even though it is capable of handling sub-daily workstation rotations (which was explicitly activated), no switching takes place. Additionally, the effective weekly working time of employees is frequently in excess of their contracts. A high effort of replanning would be required to remove these errors.

ES and PSO both performed significantly better. The best mean results were achieved with ES(1,5). The assignment plans generated with the ES(1,5) can hardly be improved upon, even with highly complex manual changes. For this reason, and because of the vast improvement over the commercial software, these plans can be regarded as very usable. Generally, the comma selection performs better than plus selection (for the same μ and λ). The comma strategy 'forgets' the parent values after each generation, which allows for a temporary deterioration of objective function values. This is helpful in escaping from a local

Table 1. Results of the heuristics with various parameter settings (averaged over 30 runs each, apart from the constructive method). Best results are bold.

heuristic	mean error	minimal error	std. dev.	job-changes	understaff. in minutes	overstaff. in minutes $d_{wt} > 0$	too much weekly work. time in min.
constructive method	84690.0	84690	-	0.0	1500.0	0.0	83190.0
PSO(20)	37117.9	14385	11808.9	389.9	834.0	20.0	35874.0
PSO(40)	40583.7	14446	12832.3	402.5	966.0	26.0	39189.2
PSO(200)	51879.5	27967	12578.4	463.9	1142.0	12.0	50261.6
ES(1,5)	**8267.1**	**5924**	1265.8	214.3	834.0	8.0	7210.8
ES(1+5)	47198.1	13720	20486.4	444.5	1870.0	28.0	44855.6
ES(10,50)	17528.5	8459	4094.2	247.3	1170.0	12.0	16099.2
ES(10+50)	49794.5	24609	26010.7	451.7	2022.0	24.0	47296.8
ES(30,200)	22222.7	13579	5780.6	283.9	1130.0	10.0	20798.8
ES(30+200)	39491.5	25706	14801.5	460.7	1720.0	12.0	37298.8

optimum. With regard to improving solutions, a tendency can be seen in the comma selection toward smaller populations. This can also be observed in PSO, where 20 particles perform best. Because of the uniform termination criterion of 400,000 fitness calculations, a smaller population or swarm means more iteration cycles. Many steps are required to arrive at a good plan. Thus, it seems preferable to track changes for more iterations as compared to richer diversity (through larger populations) of the solution space in each iteration. The effect is not so clearly visible for the ES with plus selection. A plus strategy is more easily trapped in a local optimum, which is particularly pronounced when the population is small. So there are two opposite effects, resulting in an overall medium performance of the plus strategy.

PSO(20) and ES(1,5) provided the best mean error results in their respective groups. With 30 independent runs for each heuristic it is possible to test the statistical significance of the performance difference. A Levene-test revealed the heterogeneity of variances (test level 5%) between both groups ($F = 53.17, p < 0.001$). The corresponding t-test with a 95% confidence interval confirms the better performance of ES(1,5) with a very high statistical significance ($p < 0.001$ for H_0). The overall superiority of the ES (comma selection) over PSO is in contrast to experiences reported in [7] for logistics. However, the solution space of the logistics problem contained only 36,400 decision variables compared to the 131,400 variables in the current retail application. Also, the working time models were given beforehand which significantly reduced the complexity.

8 Conclusions

Our work focuses on the meaningful practical problem of simultaneously assigning staff to workstations and generating optimised working time models on the basis of given demand. For a highly complex real-world application from retailing, it was shown that modern metaheuristics significantly outperformed a commercial constructive heuristic. The computational requirements of the metaheuristics are high. This is acceptable, though, since the task is not time-critical.

Moreover, if the planning is performed for a month instead of an entire year the computational effort would be reduced accordingly.

The evolution strategy was clearly superior over particle swarm optimisation. Particularly the ES-solutions generated with comma selection and small populations appear excellent and very usable in practice, as was also confirmed by the cooperating company. This success must be attributed to the operators of the ES since the coding of PSO and ES were identical, just as the total number of inspected solutions in each run. However, a joint consideraton of these results with the outcome on a simpler real-world staff scheduling problem from logistics as reported in [7] suggests a non-linear relation between the complexity of a search space and the success of individual metaheuristics, even if the underlying application problems are of the same general problem class. In practice it is difficult to estimate a priori which metaheuristic approach will be most successful on a given practical application. A solution to this open research question would obviously be of great practical value.

References

1. Beyer, H.G., Schwefel, H.P.: Evolution strategies - A comprehensive introduction. Natural Computing 1(1), 3–52 (2002)
2. Chu, S.C., Chen, Y.T., Ho, J.H.: Timetable Scheduling Using Particle Swarm Optimization. In: First International Conference on Innovative Computing, Information and Control, vol. 3, pp. 324–327 (2006)
3. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B., Sier, D.: An Annotated Bibliography of Personnel Scheduling and Rostering. Annals of Operations Research 127(1-4), 21–144 (2004)
4. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
5. Günther, M.: Sub-daily staff scheduling data sets and benchmarks, http://www.tu-ilmenau.de/fakww/2608+M54099f70862.0.html
6. Günther, M., Nissen, V.: A Comparison of Neighbourhood Topologies for Staff Scheduling with Particle Swarm Optimisation. In: Mertsching, B., Hund, M., Aziz, M.Z. (eds.) KI 2009. LNCS, vol. 5803, pp. 185–192. Springer, Heidelberg (2009)
7. Günther, M., Nissen, V.: Sub-Daily Staff Scheduling for a Logistics Service Provider. In: KI – Künstliche Intelligenz, vol. 25 (2010) (accepted)
8. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proc. of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
9. Prüm, H.: Entwicklung von Algorithmen zur Personaleinsatzplanung mittels ganzzahliger linearer Optimierung. Master's thesis, FH Trier (28042006)
10. Sauer, J., Schumann, R.: Modelling and Solving Workforce Scheduling Problems. In: PUK, pp. 93–101 (2007)
11. Tien, J.M., Kamiyama, A.: On Manpower Scheduling Algorithms. SIAM Review 24(3), 275–287 (1982)

Bee-Sensor: A Step Towards Meta-Routing Strategies in Hybrid Ad Hoc Networks

Israr Ullah[1], Muhammad Saleem[2], and Muddassar Farooq[1]

[1] Next Generation Intelligent Networks Research Center (nexGIN RC)
National University of Computer and Emerging Sciences (FAST-NUCES)
Islamabad, Pakistan
[2] Center for Advanced Studies in Engineering (CASE), Islamabad, Pakistan
israrullahkk@yahoo.com, msaleem@case.edu.pk, muddassar.farooq@nu.edu.pk

Abstract. In next generation ad hoc networks, MANETs and WSNs will cohesively integrate to provide a unified ad hoc framework. Such a hybrid network – due to conflicting operational environments – presents unique challenges for routing protocols. A recently proposed BeeSensor protocol inherits relevant features from BeeAdHoc – a bee-inspired protocol for mobile ad hoc networks. In this paper, we would first do requirements engineering of protocols for hybrid networks and then enhance BeeSensor with relevant features to make it suitable for MANETs and WSNs. Finally, we implement enhanced BeeSensor in famous ns-2 simulator and compare its performance with well known MANET protocols namely DSR, AODV and DSDV. The results of our experiments show that BeeSensor – using our mobility model – delivers similar or better performance compared with its competitors in low mobility scenarios. But its performance relatively degrades in high mobility scenarios. Towards the end of the paper, we propose changes that can overcome these shortcomings.

Keywords: MANETs, WSNs, meta-routing.

1 Introduction

Routing in both of types of ad hoc networks – Mobile Ad Hoc Networks (MANETs) and Wireless Sensor Networks (WSNs) – is an active area of research. A number of classical protocols – AODV [2], DSR [3] and DSDV [4] – and bio-inspired protocols – AntHocNet [6], Termite [7] and BeeAdhoc [10] – are proposed in the literature for MANETs. These protocols are eventually adapted to a WSN environment by researchers with a special focus on energy efficiency and assuming static network. In next generation ad hoc networks, MANETs and WSNs will cohesively integrate to provide a unified ad hoc framework. Such a hybrid network – due to conflicting operational environments – presents a unique challenge for routing protocols: develop meta-routing strategies. To the best of our knowledge, little work is done related to routing on hybrid ad hoc networks.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 392–399, 2010.

We follow our three step protocol engineering to systematically develop a meta routing protocol for hybrid networks: (1) analyze an existing WSN protocol, BeeSensor, that has shown promising performance in WSN environment as reported in [9], (2) port BeeSensor to ns-2 that is a well known MANET simulator, and do pilot studies to monitor its performance in mobility scenarios, and (3) enhance BeeSenor with relevant features so that its shortcomings (identified in step 2) are rectified. Using this three step methodology, we develop a meta routing strategy for hybrid networks by incorporating relevant features of BeeSensor and BeeAdHoc: (1) on demand discovery of paths, (2) using fixed length routing agents, (3) implementing source routing without the need to add a header in front of a data packet, (4) using swarms to implicitly monitor validity of wireless links, (5) exploiting paths having nodes with high energy reserves, and (6) distributing loads on multiple paths.

Organization of the Paper: The rest of the paper is organized as follows. We identify the major challenges in designing and developing a meta-routing protocol for hybrid ad hoc networks in Section 2. A brief description of BeeSensor protocol is presented in Section 3. The empirical evaluation framework used for the performance analysis of BeeSensor in MANET environment is described in Section 4. The simulation results are discussed in Section 5. The modifications to the design of BeeSensor protocol are listed in Section 6. Finally, we conclude the paper with an outlook to our future research.

2 Challenges in the Design of a Meta-Routing Protocol

The most important challenge in developing a meta-routing protocol for hybrid networks is that it must satisfy a number of conflicting requirements: (1) it must be able to cater for mobility without compromising packet delivery and delay bounds, (2) it must be able to detect path failures without the need to monitor the wireless links in a proactive manner, (3) efficient route discovery in large scale networks and timely updates of the routes (low convergence time). Currently, MANETs protocols quickly adapt routes to the continuously changing topology by monitoring wireless links in a proactive manner. Moreover, it is assumed that this control traffic will not cause congestion because the network is lightly loaded. However, this assumption is void in WSNs where large number of sensor nodes – with limited processing and battery – cannot periodically launch these probe packets; otherwise, the network would become congested and critical events would not be delivered to the sink node. To conclude, a protocol for hybrid network must not have aggressive sampling of the paths to detect path failures.

Moreover, the idea of launching periodic discovery packets – as employed in proactive protocols – is also infeasible because this results in large convergence time that makes it difficult to adapt to rapidly changing network topology. Last but not least, it results in large energy consumption as well. The challenge is: *to develop a protocol that uses relatively small number of control packets (agents) but is able to adapt to changing topology and provides relatively high performance with low energy consumption.*

We now present the first step of our protocol engineering approach in which we describe the key features of BeeSensor protocol that helps in understanding its feasibility for hybrid ad hoc networks.

3 BeeSensor as a Meta-Routing Protocol

BeeSensor is a bee-inspired multipath routing protocol for ad hoc networks which discovers routes in a reactive fashion. It is derived from BeeAdHoc protocol that makes it inherently suitable for MANETs as well. The mapping of bee colony concepts to BeeSensor is skipped here for brevity, but an interested reader is referred to [1] and [8] for details. We now describe the architecture and working of BeeSensor protocol.

3.1 Agent Model

Like BeeAdHoc, BeeSensor works with four types of agents: packers, scouts, foragers and swarms. Packers collect data packets from the transport layer and look for a suitable forager on the dance floor for its destinations. Scouts are responsible for route discovery. Foragers are the main agents that transport data packets from their sources to the respective destinations. Swarms are used to explicitly transport foragers back to the source node, if they cannot be implicitly piggy backed on the traffic from the destination towards the source (in case of UDP, where no acks are sent back to the source node).

3.2 Working of BeeSensor Protocol

Route Discovery. In BeeSensor, routes are discovered in a reactive manner. When a packet is received at the routing layer for an unknown destination, a forward scout - carrying the data packet - is launched to discover a route to the destination. Forward scouts propagate in the network using the stochastic broadcast pattern after a certain threshold hop limit. At each intermediate node, a reward value is computed using:

$$Reward = \frac{MinNodeEnergy}{Hops}. \tag{1}$$

Forward scouts - on reaching the destination - travel back to the source as backward scouts. Backward scouts use the reward values stored at the intermediate nodes to select the next hop nodes leading to the source node. Generally, they prefer paths with high rewards.

Recruitment of agents. Once a backward scout arrives at the source it is guaranteed that a route, identified with a unique pathID (generated by the destination), is available for data transport. The scouts are then forwarded to the dance floor, where they mimic the process of recruiting agents by computing a dance number as given below:

$$DN_{pid} = \left\lceil \frac{|(\beta - (E_{max} - E^r_{pid}))|}{\gamma} \times \alpha(e) \right\rceil, \tag{2}$$

where E^r_{pid} is the minimum remaining energy of the path (identified by a unique ID i.e. pid) reported by the backward scout, E_{max} is the initial (maximum) energy level, $\alpha(e)$ is a function of the number of events waiting in the cache and β, γ are user defined constants. The Path's quality is computed using:

$$Q_{pid} = \frac{DN_{pid} * Min.Energy_{pid}}{HopCount_{pid}}. \tag{3}$$

Equation (3) is then used to compute the probability of selecting a path:

$$P_{pid} = \frac{Q_{pid} * 100}{\sum_{i=1}^{k} Q_i} \tag{4}$$

where k is the total number available paths for a specific destination, Q_{pid} is the path's quality, and P_{pid} is the percentage relative probability for $Path_{pid}$. Equation (4) favors the shortest paths; however, in case if paths are of equal length, the paths with high remaining energy is preferred. In Equation (3), DN_{pid} is included to ensure proportional use of all discovered routes resulting in depleting battery of different sources at the same rate. After making these calculations, an entry is made into the routing table.

Data Forwarding. Once the routes are discovered, foragers are selected probabilistically - to ensure load balancing that maximizes the network lifetime - to carry data packets from a source to a destination. At intermediate nodes, foragers are forwarded on the basis of their pathIDs. At destination, data is passed to the upper layers and foragers wait for a specific interval of time so that the destination node can piggy back them on the traffic from the destination to the source. If it does not happen, a swarm agent is generated that explicitly carries them to their source node. As a result, the source nodes always have foragers that represent valid routes to the destination.

Handling Link failures. Unlike AODV and DSR, no explicit control packets are used in BeeSensor for handling link failures. If no foragers are available for a destination, and none of them comes back within a specified period of time, the associated path is assumed to have become invalid (because of broken links) and the corresponding entry is removed from the routing tables. In this case, a fresh scouting process is initiated. In this way, link failures are implicitly detected without the need of aggressively sampling the links through control packets.

4 Empirical Evaluation Framework

In the second step of our protocol engineering approach, we implement BeeSensor in a well known simulator – ns-2 – and evaluate its performance in a mobility scenario. The scenario consists of 50 nodes moving in a rectangular area of $1500 \times 300m^2$ with varying speed using Random Way-Point mobility modal. All flows simulate peer to peer communication using UDP protocol at the transport layer protocol. We assume CBR sources generating packets at a specified rate.

We use the default settings of ns-2 for the transmission range of a node, network's bandwidth and the battery model.

We compare the performance of BeeSensor with that of AODV, DSR and DSDV using four different metrics. *Packet delivery ratio* is the ratio between received and sent packets by all nodes. *Latency* is the difference (average) between a packet receiving and sending time. *Normalized routing load* is the ratio between total number of control and delivered data packets. *Energy efficiency* is the energy consumed per kilobyte data transportation. We collect the values of these metrics by changing moving speed of nodes, the number of flows and the size of packets. Each experiment is run for a duration of 1000 seconds and the reported results are an average of 10 independent runs.

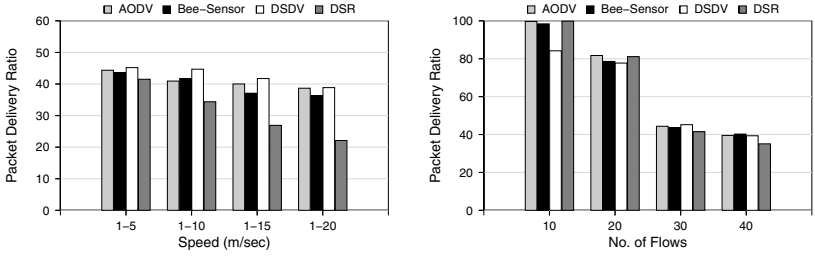
5 Discussion on Results

5.1 Packet Delivery Ratio

The packet delivery ratios of the protocols once we change speed and the number of flows are shown in Figure 1. It is interesting to see that the packet delivery ratio of all protocols is alarmingly low (40%) in most of the cases. This poor performance is attributable to UDP that does not guarantee a reliable delivery of packets. The DSR especially fails at high speeds and the reason for this is its use of stale entries (route caches) that do not represent the latest topology of the network (especially in high speed scenarios). Moreover, as we increase the number of flows to 30, the packet delivery ratio of all protocols drop to 40% because of congestion in the network.

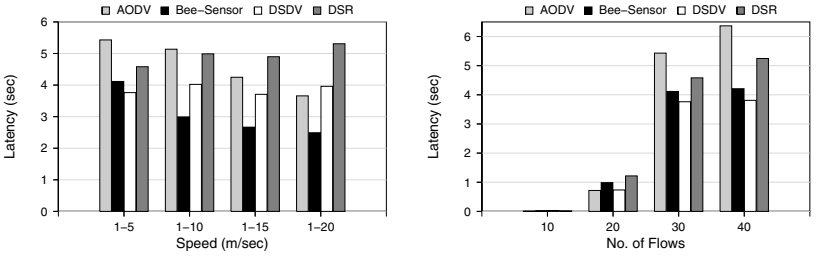
5.2 Latency

The corresponding latency values are plotted in Figure 2 by varying the speed and the number of flows. In most of the cases, BeeSensor achieves the lowest latency among all protocols. The second important observation is that latency increases because of network congestion (due to increased number of flows). The



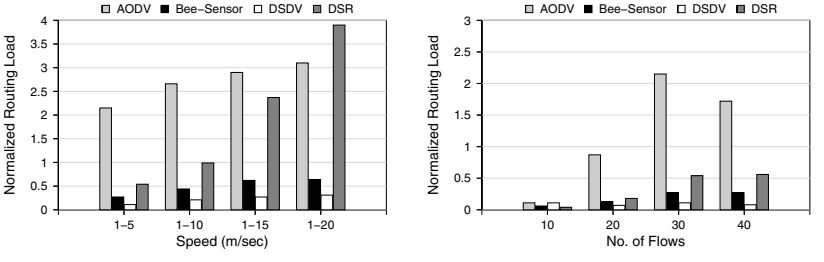
(a) PDR Vs. Speed (flows: 30) (b) PDR Vs. Flows (speed: 5m/sec)

Fig. 1. Packet delivery ratio (PDR) against varying speed and the number of flows



(a) Latency Vs. Speed (flows: 30) (b) Latency Vs. Flows (speed: 5m/sec)

Fig. 2. Latency against varying speed and the number of flows



(a) NRL Vs. Speed (flows: 30) (b) NRL Vs. Flows (speed: 5m/sec)

Fig. 3. Normalized routing load (NRL) against varying speed and the number of flows

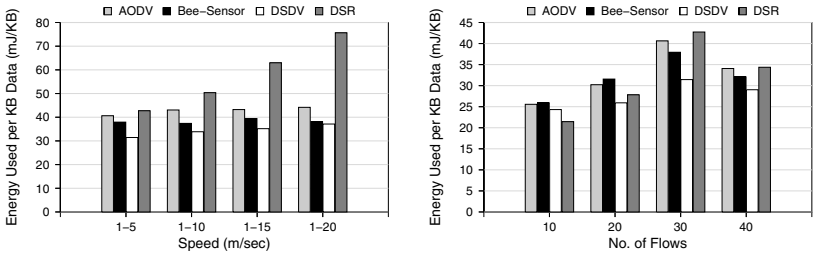
reason is that queues on intermediate nodes build up and packets have to wait longer in the queue before their transmission. The average delay of BeeSensor remains unaffected with an increased in the packet size (the results are skipped for the sake of brevity).

5.3 Normalized Routing Load

The normalized routing load for the protocols are shown in Figure 3. The normalized routing load of BeeSensor is relatively small and comparable to DSDV in all cases. AODV and DSR generate too much of control packets due to their aggressive policy; as a result, they create instability in the network. On the other hand, BeeSensor – though a reactive protocol like AODV and DSR – not only reacts to the changes in the topology but also generates less traffic. The important reason is that stochastic flooding in BeeSensor is responsible for small routing load. (AODV has the largest overhead cases). It is interesting that a pure proactive protocol has a small overhead compared with reactive protocols. The reason is that if the topologies change continuously under high load network conditions, proactive discovery is not a bad option.

5.4 Energy Efficiency

Energy consumption is directly proportional to the number of transmissions in the network. The same trend can be seen in Figure 4 which shows the energy-efficiencies of the protocols. DSDV with lower routing load has better energy efficiency. On the other hand, the energy consumption of AODV and DSR (as expected) in high speed scenarios is relatively large. Energy efficiency is an important consideration in WSNs and hence a protocol is preferred if it optimally uses the battery of sensor nodes.



(a) Energy Efficiency Vs. Speed (flows: 30)

(b) Energy Efficiency Vs. Flows (speed: 5m/sec)

Fig. 4. Energy efficiency against varying speed and the number of flows

6 Modifications to Original Design of BeeSensor

In the previous section, we see that BeeSensor's performance degrades in high mobile scenarios. In this section, we propose changes (the step 3 of our protocol engineering) to the design of BeeSensor which can improve its performance even in high mobility scenarios:

1. The waiting time of foragers at the destination node results in slow sampling of discovered paths. If the swarm size is made a function of speed (it decreases with an increase in speed and vice versa), then BeeSensor can adapt its response time with the network dynamics. This may lead to an increased energy consumption, but we have to make this compromise.
2. BeeSensor can make use of the underlying MAC layer to determine the frequency of link failures. If this frequency is high (high mobility case), the timeout function of foragers can be adjusted to have smaller route expiry times. More specifically, we can make the route expiry time a function of network characteristics as observed by the local nodes. This can also lead to a reduced response time improving its performance in high mobility scenarios.

7 Conclusions and Future Work

Meta-routing strategies will become an important paradigm for routing in next generations networks (3G and 4G networks). In this paper, we address some of the issues related to routing in hybrid ad hoc networks. We follow a three step protocol engineering to adapt BeeSensor for both MANETs and WSNs. In future, we intend to validate our proposed improvements in BeeSensor by studying its performance in high mobility scenarios.

References

1. Farooq, M.: Bee-Inspired Protocol Engineering: from Nature to Networks. Natural Computing Series. Springer, Heidelberg
2. Perkins, C.E., Royer, E.M.: Ad hoc on demand Distance Vector routing. In: 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 1999), pp. 90–100 (1999)
3. Johnson, D., Maltz, D.A.: Dynamic source routing in ad hoc wireless networks. In: Imielinski, T., Korth, H. (eds.) Mobile Computing. Kluwer Acad. Publ., Dordrecht (1996)
4. Perkins, C.E., Bhagwat, P.: Highly dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for mobile computers. ACM Computer Communication Review 24(4), 234–244 (1994)
5. Rajagopalan, S., Shen, C.: ANSI: A swarm intelligence-based unicast routing protocol for hybrid ad hoc networks. J. Syst. Archit. 52(8), 485–504 (2006)
6. Di Caro, G., et al.: AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. Europ. Trans. on Telecommunications V16, 443–455 (2005)
7. Roth, M.H.: Termite: a Swarm Intelligent Routing Algorithm for Mobile Wireless Ad-Hoc Networks. Doctoral Thesis. UMI Order Number: AAI3162791. Cornell University (2005)
8. Saleem, M., Farooq, M.: BeeSensor: A bee-inspired power aware routing protocol for wireless sensor networks. In: Giacobini, M. (ed.) EvoCOMNET 2007. LNCS, vol. 4448, pp. 81–90. Springer, Heidelberg (2007)
9. Saleem, M., Farooq, M.: A framework for empirical evaluation of nature inspired routing protocols for wireless sensor networks. In: IEEE CEC 2007, pp. 751–758 (2007)
10. Wedde, H.F., et al.: Beeadhoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior. In: GECCO 2005, pp. 153–160 (2005)

Cooperation in a Heterogeneous Robot Swarm through Spatially Targeted Communication

Nithin Mathews[1], Anders Lyhne Christensen[2],
Rehan O'Grady[1], and Marco Dorigo[1]

[1] IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{nmathews,rogrady,mdorigo}@ulb.ac.be
[2] Instituto de Telecomunicações, Lisbon, Portugal
anders.christensen@iscte.pt

Abstract. We consider a heterogeneous swarm robotic system composed of wheeled and aerial robots called foot-bots and eye-bots, respectively. The foot-bots are able to physically connect to one another autonomously and thus form collective robotic entities. Eye-bots have a privileged overview of the environment since they can fly and attach to metal ceilings. In this paper, we show how the heterogeneous swarm can benefit from cooperation. By using so-called *spatially targeted communication*, the eye-bot is able to communicate with selected groups of foot-bots and instruct them on how to overcome obstacles in their path by forming morphologies appropriate to the obstacle encountered. We conduct experiments in simulation to quantify separately the benefits of cooperation and of spatially targeted communication.

1 Introduction

We use a heterogeneous swarm robotic system consisting of two types of robots: foot-bots and eye-bots (see Fig. 1a and Fig. 1b). The foot-bots are capable of autonomous self-assembly which means that they can make physical connections with one another and form collective robotic entities. In this paper, we focus on the task of navigating through an environment that contains a gap. Depending on the width of the gap, the foot-bots may need to self-assemble into a collective robotic entity to successfully overcome the gap.

In a previous study [10], a team of wheeled robots autonomously self-assembled into different morphologies to solve different tasks, one of which was a gap crossing task similar to the one considered in this paper. In that study, however, the solution to each task was preprogrammed. For example, the wheeled robots did not have the sensory capabilities to estimate the width of a gap. Therefore, on encountering a gap, they would always self-assemble into a four robot line morphology irrespective of the width of the gap. In this paper, we present an approach for cooperation between aerial and wheeled robots that enables self-assembling robots to adaptively generate appropriate morphologies to a priori unknown tasks.

In the task we consider, the heterogeneous swarm is located in an environment consisting of a start zone, a target zone, and a gap that separates the two zones

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 400–407, 2010.

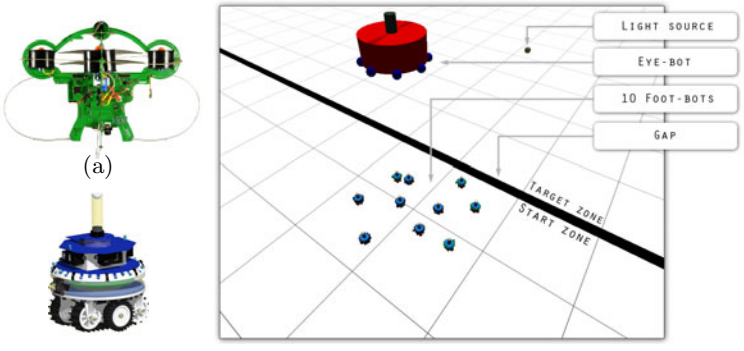


Fig. 1. The heterogeneous swarm robotic system and the task considered in this study. (a) The prototype of the eye-bot. (b) A CAD model of the foot-bot. Both robot types are being developed at EPFL within the framework of the Swarmanoid project. More information about the project is available at http://www.swarmanoid.org. (c) A depiction of the task considered. The dark strip represents the gap which separates the arena into a start zone and a target zone. The circular object shown in the target zone is the light source. An eye-bot and 10 foot-bots are visible in the start zone.

(see Fig. 1c). A light source perceivable by the foot-bots is located in the target zone. At the start of each experiment, 10 foot-bots are placed at random positions with random orientations within a square area of 2 m x 2 m in the start zone. The foot-bots use their light sensors to detect and drive to the light source in the target zone. They use the ground sensors to avoid falling into the gap. An eye-bot is assumed to be attached to the ceiling in the start zone using its system of magnets. It is able to perceive all the foot-bots in the start zone. The eye-bot can estimate the width of the gap by using its pan-and-tilt camera and the on-board image processing software. To reach the target zone, the foot-bots may need to connect to each other to form a collective morphology, such as a line morphology [2]. Note that the minimal length of such a line morphology (i.e., the number of foot-bots in the line) that guarantees a safe crossing of the gap depends on the width of the gap. In this study, we vary the width of the gap between 5 cm, 10 cm, 15 cm and 25 cm. These different gap widths require the foot-bots to form a line morphology of 1, 2, 3 and 4 foot-bots respectively. The task is considered to be completed when the final foot-bot of the line morphology has crossed the gap and reached the target zone.

To enable cooperation in the heterogeneous swarm, we use a combination of techniques developed in previous research. Firstly, the eye-bot establishes *spatially targeted communication* [9] with a selected group of foot-bots. Secondly, the eye-bot sends morphology growth instructions to these foot-bots in the form of SWARMORPH-script [2] instructions. Both spatially targeted communication and SWARMORPH-script have been successfully tested on real robotic hardware

in previous studies, see [9] and [2], respectively. Our approach does not require any form of global information.

At the time of writing, the heterogeneous swarm robotic system is still under development. We therefore use a custom physics-based simulator named ARGoS [13] to study separately the benefits of cooperation between the two robot types and spatially targeted communication.

2 Related Work

Most previous studies on cooperation in heterogeneous systems have focused on tightly-coupled heterogeneous teams, see for instance [12,18,4]. In these and other similar multirobot systems, researchers have used communication and/or localization modalities such as wireless ethernet [17], infrared [7] or ultrasound [15]. In this study, we consider a heterogeneous swarm robotic system composed of numerous wheeled and aerial robots, see for instance [14]. We use on-board LEDs and cameras for communication between the foot-bots and the eye-bots. The foot-bots can also communicate with each other using a communication system based on infrared and radio [16].

Many researchers have designed and studied systems that can reconfigure or self-assemble into physically connected structures [6]. To date, several hardware architectures for self-propelled self-assembling robotic systems have been proposed and implemented [3,5]. In this study, we use foot-bots which are self-propelled, fully autonomous and can self-assemble. The performance benefits of different self-assembly strategies for similar robots has been studied previously [11], however the study was conducted on a homogeneous system and morphology control was not considered. For self-assembling and self-reconfigurable systems, several different control approaches have been proposed [1,8]. In this work, we use a language called SWARMORPH-script [2] that allows target morphologies to be described as distributed control logic.

3 Methodology

We achieve cooperation using two mechanisms developed in previous research. Firstly, the eye-bot establishes a *spatially targeted communication* [9] link with a group of foot-bots that is appropriately located (i.e., near the gap) and has an appropriate size (i.e., the precise number of foot-bots required to cross the gap). Secondly, the eye-bot instructs these selected foot-bots to form the line morphology (i.e., target morphology) that will allow them to cross the gap [2].

In [9], a LEDs and camera-based communication is used by a robot to first narrow down the number of potential recipients of a broadcast message to a single *seed* robot. This one-to-one communication link is then expanded to include the closest neighbors of the seed robot such that a one-to-many commnication link of a desired size can be created. Such a dedicated communication link enables the eye-bot to ensure that subsequently broadcasted messages will only be processed

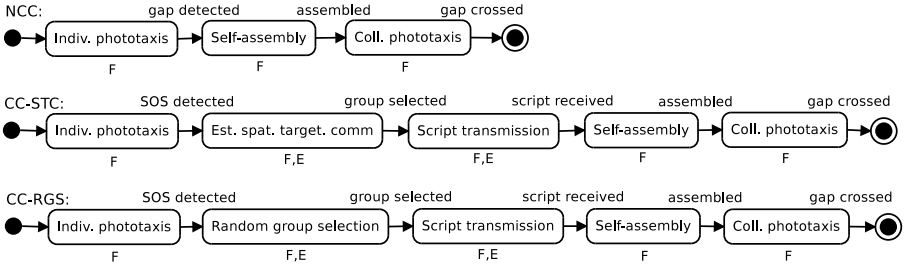


Fig. 2. Decomposition of control strategies into phases. Phases only involving foot-bot are marked 'F', phases involving foot-bot eye-bot cooperation are marked 'F.E'. i) NCC: non-cooperative control, ii) CC-STC: cooperative control with spatially targeted communication and iii) CC-RGS: cooperative control with random group selection. NB 'Indiv. phototaxis' = 'Individual phototaxis', 'Coll. phototaxis' = Collective phototaxis, 'Est. spat. target. comm.' = 'Establishing Spatially Targeted Communication'.

by the selected group of foot-bots even though other foot-bots may also be able to receive the messages.

We use such dedicated communication links to let the eye-bot send instructions to the foot-bots on how to self-assemble into the target morphology. These instructions are sent in SWARMORPH-script [2]. SWARMORPH-script is a language for distributed self-assembly and morphology control for autonomous self-assembling robots. The eye-bot uses a protocol based on LEDs and camera to send the SWARMORPH-script required to generate the target morphology. Each foot-bot that receives such a SWARMORPH-script can execute this received control logic. In this manner, the foot-bots do not need to have any a priori knowledge about possible morphologies required or even possible tasks.

4 Experiments and Results

We ran simulation-based experiments using three different control strategies of the heterogeneous swarm. For each combination of gap size and control strategy, we ran 100 repetitions. By comparing the task completion times of the three strategies, we first analyze the benefits of cooperation through spatially targeted communication, and then isolate the benefits of spatially targeted communication. Videos of the experiments conducted are available online at: http://iridia.ulb.ac.be/supp/IridiaSupp2010-007/.

4.1 The Three Control Strategies

The three strategies are presented in Fig. 2. The simplest strategy is **NCC** — non-cooperative control. In this strategy, the foot-bots operate without cooperating with the eye-bot. They initially move towards the light and form a four robot line morphology when they encounter any gap (irrespective of the width

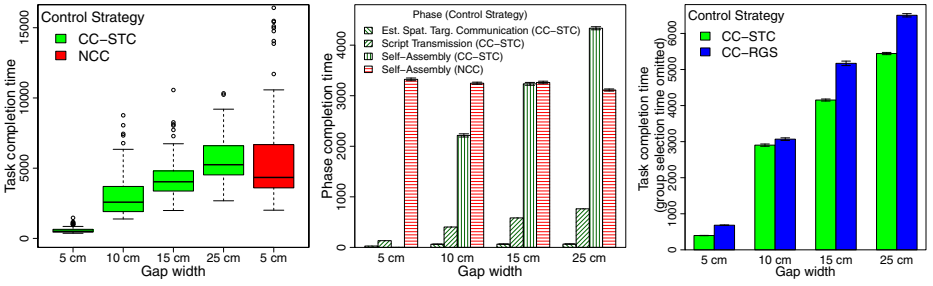


Fig. 3. Results of the experiments showing task/phase completion times in simulation steps. In Fig. 3b and 3c, the whiskers represent the standard deviation. (a) Box-and-whisker plot comparing CC-STC and NCC for varying gap widths. (b) Bar-plot showing a breakdown of the time spent in different phases of CC-STC and NCC. (c) Completion times of CC-STC and CC-RGS minus the time taken to form the group.

of the gap). The foot-bots are pre-loaded with the SWARMORPH-script instructions to form the morphology and cross the gap by performing collective phototaxis.

The methodology presented in this work is implemented in **CC-STC** — cooperative control with spatially targeted communication. In this strategy, the foot-bots initially move towards the light until the eye-bot initiates the process to establish a spatially targeted communication link with the minimal number of foot-bots required to form the target morphology. The communication link is established with foot-bots that are favorably located (i.e., close to the gap) to solve the task. Subsequently, a SWARMORPH-script is sent to these foot-bots. Once the target morphology is generated, the foot-bots perform collective phototaxis to cross the gap.

The final strategy is **CC-RGS** — cooperative control with random group selection. This strategy allows us to isolate the performance benefits of spatially targeted communication. The strategy is identical to the CC-STC strategy, except that instead of selecting the foot-bots to form the target morphology on the basis of their favorable location, the eye-bot randomly selects the minimal number of foot-bots required to form the target morphology.

4.2 Benefits of Cooperation in the Heterogeneous Robot Swarm

We compare the task execution times of strategies NCC and CC-STC to analyze the benefits of cooperation through spatially targeted communication. The results are shown in Fig. 3a. In the case of NCC, we have only plotted the results of the narrowest gap (5 cm), as the task completion times between the various gap widths did not prove to be significantly different for the NCC strategy.

According to the results in Fig. 3a, the median task completion times of CC-STC are 507, 2590 and 4032 simulation steps for gaps of width 5, 10 and

15 cm, respectively. This means that CC-STC was 88%, 40% and 7% faster when compared to the median task completion value of NCC (4340 simulation steps). This is due to the fact that in CC-STC, the length of the line is optimal with respect to the width of the gap. However, for the widest gap (25 cm), NCC is shown to be faster than CC-STC. Intuitively, this could have been expected given that both control strategies form a line of four robots close to the gap, but in the case of CC-STC, instructions have to be first received from the eye-bot before the self-assembly process can start and therefore requires more time. Results also show that NCC has several outlier trials that take very long to complete. This is because in the NCC strategy all foot-bots in the experiment are allocated to construct the morphology and some non-connected foot-bots can interfere (sometimes severely) with the collective phototaxis of the complete morphology.

In Fig. 3b, a breakdown of the time spent in the different phases of each control strategy is shown: (i) establishing spatially targeted communication (CC-STC), (ii) transmitting the SWARMORPH-script (CC-STC) (iii) self-assembly (CC-STC), (iv) self-assembly (NCC). The results show that with the increasing size of the morphology, and therefore with the increasing length of the SWARMORPH-script that has to be transmitted, the transmission time increases. However, this communication overhead of CC-STC would become negligible if a communication modality with higher bandwidth (such as WiFi) was used. The results also show that when a line of equal length is formed in both control strategies, as in the case of 4 foot-bots, the self-assembly process of CC-STC requires on average 39% more time than that of NCC. This can be explained by the fact that in NCC there are more robots attempting to connect to a connection-inviting foot-bot which in turn leads to faster morphology formation. On the other hand, CC-STC deals with the resources optimally by only allocating precisely the required number of robots needed for the self-assembly process. The decision involving this trade-off between faster morphology formation times and optimal resource allocation may depend on the task and/or the priorities of the system.

4.3 Benefits of Spatially Targeted Communication

To isolate the benefits of spatially targeted communication, we compare the task completion times of strategies CC-STC and CC-RGS. Note that both control strategies select the seed foot-bot using the same technique. However, the selection of further foot-bots required in the target morphology is different. Therefore, in order to maintain objectivity in the comparison, in this set of experiments the time spent to select the non-seed foot-bots was omitted for both control strategies. The results are plotted in Fig. 3c.

As the results show, CC-STC was on average faster than CC-RGS independent of the width of the gap. This is because a morphology formed next to the gap require less time to reach and cross the gap than a morphology formed at a random place in the environment. We expect that this difference in terms of task completion time would be even greater for larger start zones.

Additionally, we also studied the difference in completion times between CC-STC and CC-RGS in the presence of obstacles: the foot-bots were placed in the start zone within an area of 2 m x 2 m surrounded by walls on three sides to adjoin the gap on the fourth side. We found that the presence of the walls had no significant impact on the completion time of CC-STC in which the eye-bot selects the seed and the group in favorable locations (i.e., always close to the gap and away from the walls). For the CC-RGS control strategy, on the other hand, the presence of walls had a significant negative impact on performance. When the randomly selected seed (which initiates the morphology growth process) happened to be located close to one of the walls, it could be difficult or even impossible for other foot-bots to physically connect to the seed. As a result, the task was not solved in our experiments with the CC-RGS control strategy in 13%, 29% and 34% of the experiments for the line morphology composed of 2 foot-bots, 3 foot-bots and 4 foot-bots, respectively.

5 Conclusions and Future Work

In this paper, we have demonstrated how aerial robots and wheeled robots can cooperate to solve different instances of a gap crossing task in an adaptive manner. Compared to a non-cooperative strategy, the cooperative strategy was shown to be more efficient in terms of resource allocation as the aerial robot recruited only the necessary robots based on the width of a gap. Furthermore, the cooperative strategy led to faster task completion times in the environment in which fewer than four connected robots could cross the gap. We also demonstrated the benefits of spatially target communication. When the aerial robot selected wheeled robots based on their location and based on their mutual proximity to each other, the time required to self-assemble and to cross the gap was lower than when robots were randomly selected.

Our short-term goal is to repeat the experiments shown in this paper on real robotic hardware. In ongoing research, we are investigating other cooperation mechanisms between aerial and wheeled robots, in particular where the cooperation is more bidirectional. In this study, the wheeled robots passively received instructions from the aerial robots. In the future, wheeled robots on the ground could ask an aerial robot to find additional robots for a given task, and multiple aerial robots could allocate and share groups of wheeled robots dynamically.

Acknowledgements. This work was supported by the SWARMANOID project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission, under grant IST-022888, and by the VIRTUAL SWARMANOID project funded by the Fund for Scientific Research F.R.S.-FNRS of Belgium's French Community. The information provided is the sole responsibility of the authors and does not reflect the European Commission's opinion. The European Commission is not responsible for any use that might be made of data appear- ing in this publication. Marco Dorigo acknowledges support from the Belgian F.R.S.-FNRS, of which he is research director.

References

1. Butler, Z., Kotay, K., Rus, D., Tomita, K.: Generic decentralized control for lattice-based self-reconfigurable robots. Int. Jour. of Rob. Res. 23(9), 919–937 (2004)
2. Christensen, A.L., O'Grady, R., Dorigo, M.: SWARMORPH-script: A language for arbitrary morphology generation in self-assembling robots. Swarm Intelligence 2(2-4), 143–165 (2008)
3. Damoto, R., Kawakami, A., Hirose, S.: Study of super-mechano colony: concept and basic experimental set-up. Adv. Robotics 15(4), 391–408 (2001)
4. Dias, M.B., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: A survey and analysis. Proc. of the IEEE 94(7), 1257–1270 (2006)
5. Fukuda, T., Buss, M., Hosokai, H., Kawauchi, Y.: Cell structured robotic system CEBOT: Control, planning and communication methods. Rob. and Auton. Syst. 7(2-3), 239–248 (1991)
6. Groß, R., Dorigo, M.: Self-assembly at the macroscopic scale. Proc. of the IEEE 96(9), 1490–1508 (2008)
7. Gutiérrez, A., Campo, A., Dorigo, M., Amor, D., Magdalena, L., Monasterio-Huelin, F.: An open localization and local communication embodied sensor. Sensors 8(11), 7545–7563 (2008)
8. Klavins, E., Ghrist, R., Lipsky, D.: A grammatical approach to self-organizing robotic systems. IEEE Trans. on Autom. Cont. 51(6), 949–962 (2006)
9. Mathews, N., Christensen, A.L., Ferrante, E., O'Grady, R., Dorigo, M.: Establishing spatially targeted communication in a heterogeneous robot swarm. In: 9th Int. Conf. on Auton. Agents and Multiagent Syst. (AAMAS 2010), pp. 939–946. ACM, New York (2010)
10. O'Grady, R., Christensen, A.L., Pinciroli, C., Dorigo, M.: Robots autonomously self-assemble into dedicated morphologies to solve different tasks (extended abstract). In: 9th Int. Conf. on Auton. Agents and Multiagent Syst. (AAMAS 2010), pp. 1517–1518. ACM, New York (2010)
11. O'Grady, R., Groß, R., Christensen, A.L., Dorigo, M.: Self-assembly strategies in a group of autonomous mobile robots. Auton. Robots 28(4), 439–455 (2010)
12. Parker, L.: ALLIANCE: an architecture for fault tolerant multirobot cooperation. IEEE Trans. on Rob. and Autom. 14(2), 220–240 (1998)
13. Pinciroli, C.: The Swarmanoid Simulator. Tech. Rep. TR/IRIDIA/2007-025, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2007)
14. Pinciroli, C., O'Grady, R., Christensen, A.L., Dorigo, M.: Self-organised recruitment in a heterogeneous swarm. In: 14th Int. Conf. on Adv. Rob. (ICAR 2009). Proceedings on CD-ROM, paper ID 176, p. 8 (2009)
15. Rivard, F., Bisson, J., Michaud, F., Létourneau, D.: Ultrasonic relative positioning for multi-robot systems. In: IEEE Int. Conf. on Rob. and Autom., pp. 323–328. IEEE Press, Piscataway (2008)
16. Roberts, J.F., Stirling, T.S., Zufferey, J.C., Floreano, D.: 2.5d Infrared Range and Bearing System for Collective Robotics. In: IEEE/RSJ Int. Conf. on Int. Rob. and Syst. (IROS 2009). IEEE Press, Piscataway (2009)
17. Stentz, A.T., Kelly, A., Herman, H., Rander, P., Amidi, O., Mandelbaum, R.: Integrated air/ground vehicle system for semi-autonomous off-road navigation. In: AUVSI Unmanned Syst. Symp. (2002)
18. Sukhatme, G., Montgomery, J., Vaughan, R.: Experiments with aerial-ground robots. In: Robot Teams: From Diversity to Polymorphism, pp. 345–367. AK Peters, Wellesley (2001)

Early-Stage Diagnosis of Endogenous Diseases by Swarms of Nanobots: An Applicative Scenario

Paolo Amato[1], Massimo Masserini[2],
Giancarlo Mauri[1], and Gianfranco Cerofolini[3]

[1] DISCo, University of Milano–Bicocca, Milano, Italy
{paolo.amato,mauri}@disco.unimib.it
[2] Department of Experimental Medicine, University of Milano–Bicocca, Milano, Italy
massimo.masserini@unimib.it
[3] CNISM and Department of Materials Science,
University of Milano–Bicocca, Milano, Italy
gianfranco.cerofolini@mater.unimib.it

Abstract. The development of artificial devices (*nanobots*), working as blood white cells but addressed to the recognition and eventually the destruction of endogenous pathological states, is an ambitious goal. Swarm intelligence can be a key element to successfully tackle the challenges posed by this goal. Here we describe an applicative scenario, based on swarm of nanobots, by sketching the environment in which the nanobots operate, the constraints related to their physical implementation, and the tasks they have to tackle. In this scenario, we propose to use collisions between nanorobots as a way of communication inside the swarm.

Keywords: Swarm Intelligence, Nanorobotics, Nanotechnology, Fractals, Medicine.

1 Introduction

An ambitious long-term goal of medicine is to make analyses and deliver drugs selectively at cell level [16,7]. The basic idea sustaining this goal is inspired by the *immune system*. The immune system is a heterogeneous collection of relatively homogeneous families of specialized cells spread throughout the entire organism and devoted to the surveillance, recognition, and termination of hostile guests. That the immune system is a fantastic machine for surveilling and fighting against exogenous diseases is too well known to deserve discussion. However, it is not perfect. A drawback of this system is the poor recognition of endogenous pathological cells like those responsible for cancer or self-immune diseases. In this context it should be interesting to develop artificial devices (*nanorobots*) working as blood white cells although addressed to the recognition of endogenous pathological states only.

A nanorobot (since now on *nanobot*) is any artificial machine with overall size of the order of a few micrometers or less in all spatial directions, constituted by nanoscopic components with individual dimensions in the interval $1-10^2$ nm [14].

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 408–415, 2010.

A major challenge is the design of nanobots (i) addressed to monitoring *in vivo* the health state of complex systems like *homo sapiens sapiens*; (ii) able to embed sophisticated functions, like navigation, recognition, and data transmission; and (iii) suitable for being manufactured by processes compatible with today and likely tomorrow semiconductor industries.

A route to reach this goal has been described by the authors in [6] and popularized in the website [5]. In tackling these challenges it is important to take into account that, in view of their limited size, nanobots are necessarily devices with very limited computational resources and that their interaction can only happen locally. Although at first sight these facts seem to make the challenges really daunting, from nature we have experiences of tiny living beings (like ants) with very limited resources but nonetheless able to perform complex tasks. In other words, swarm intelligence [1] seems to be a key tool for designing nanobot control algorithms. In fact, ideas regarding swarm of micro- and nano-bots for medical applications have already been presented in literature [11,13,14]. In this work we want to stimulate the discussion on how swarm intelligence can be exploited for the early-stage diagnosis of endogenous diseases, by sketching the environment in which the nanobots operate, the constraints related to their physical implementation, and the tasks they have to tackle.

2 *In vivo* Monitoring at Circulatory-System Level

As far as the complete and continuous surveillance of the entire molecular pattern of an organism seems beyond any current possibility, the first problem is the identification of the appropriate surveillance level: genes, cells, or blood. Among them, the level that seems to pose less technological issues is the blood one [6]. Consequently we focus on the idea of injecting nanobots in the circulatory system. The circulatory system can be described as a complex tree forming a double canopy inside each organ. The root of this tree is the aorta and the leaves are the capillaries. Since the capillaries feed the cells, every cell is within a diffusion length from a capillary. The above argument implies that a nanobot in a capillary could feel the metabolic pattern of the family of cells fed by the capillary itself, thus surveying the cells contained within a diffusion length.

Fractal-tree model of the blood circulatory system. The design of an efficient algorithm to control the swarm of nanobots requires an adequate model of the circulatory system. Since the work of Mandelbrot, the fractal geometry of biological structures [10] has suggested that fractal methods could be used for modeling the human circulatory system. In fact, in the recent years the structure of blood vessel in any part of human body has been described in fractal terms. A fractal tree can be loosely defined as a trunk and a number of branches, each one looking like the tree itself, thus creating a self-similar object. Often, these appear strikingly similar to real trees. From the analysis in [8], it seems that the tree structure is asymmetrical at the beginning (i.e., at aorta level), becomes continuously more symmetrical with almost every branching level, and ends up in full symmetry in capillary bed.

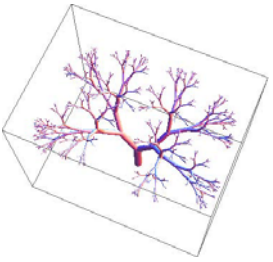


Fig. 1. 3D geometry of fractal vascular symmetrical tree

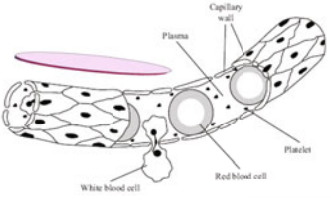


Fig. 2. Pictorial view of a nanobot entering a capillary in the vicinity of a neuron (roughly on scale). Neuron image by Nicolas Rougier.

Basic analysis of the fractal-tree model. In [6] it is estimated that there are around 2^{35} capillaries, that their diameter is $\simeq$ 5μm, and that their length is in the range $160 - 320$ μm. A cross-validation of the above estimate can be given by considering the tree model of the circulatory system. The relation between blood vessels diameters at branching nodes is given by the equation $d^{\Delta} = d_l^{\Delta} + d_r^{\Delta}$, where d is the diameter of the parent segment and d_l, d_r of the children segments, and Δ is the diameter exponent [10,12]. For the sake of simplicity, let's suppose that the fractal tree is symmetric. By using as constant diameter exponent $\Delta = 2.7$ (the suggested value for human arterial blood vessel [8]), the ratio between parent and children diameter is 0.77. Then by assuming as starting point an aorta diameter $d_1 \simeq 3.5$ cm (corresponding to the considered aorta cross section A_1), after 35 branchings the diameter of the last capillaries (the leaves of the tree) is $\simeq$ 5μm; a result consistent with the one obtained with the previous estimates. Figure 1 shows a symmetrical fractal tree (whose depth is 9) with the above parameters. Also the formula for the rotation angle ϕ is given in [12]; here $\phi \simeq \frac{\pi}{5}$.

3 Swarm of Nanobots for Surveilling the Organism

The surveillance system considered in this work is constituted by two parts: a *central unit*, externally accessible but permanently resident in the organism (e.g., as an earring), and a *swarm of nanobots*. In Ref. [4] it was hypothesized that each nanobot is a self-propelled machine, taking energy from the environment, able to recognize and dock the target cell, to sense its membrane and neighborhood, to recognize its health state, to store the information, to transfer it to the central unit, and eventually (once allowed) to destroy the malignant cell. Today a swarm so done (actually an *auxiliary immune system*) seems beyond current possibilities; we now believe that something similar can be achieved by specializing the agents of the swarm to diagnosis (*scouts*) or to therapy (*workers*). A roadmap for nanobot diagnostics can since now be defined; more difficult is to imagine a general framework for the use of nanobots in therapy.

A scout is nothing but a circulating nanolaboratory for blood analysis *in situ*. The swarm may in turn be formed by sub-swarms, each addressed to set of metabolites characteristic of the target tissue. The population of the various sub-swarms must be tuned to have an optimal surveillance of the organism. The advantage of the scout swarm over conventional blood analysis is obvious: even a non-invasive swarm of 10^6 scouts guarantees that each capillary is checked more than ten times per month and the detection of a pathological marker localizes its emission from a small number, on the mean 6×10^3, of cells. Moreover, the concentration resulting from the release of a given amount of markers in a capillary (of volume $\Omega_{\bar{n}} = 2.5 - 5 \times 10^{-7}$ cm^3) is reduced after dilution in the entire circulatory system (of volume $\Omega = 5 \times 10^3$ cm^3) by 10 orders of magnitude. At the present stage of knowledge, the hypothesized swarm is certainly far from being producible, but it is not an (irrational) dream because most of the critical steps required for its preparation have already been established.

A nanobot can be endowed with a simple control system, however allowing it to master its numerous and sophisticated functions (sensing, recognizing chemical patterns, controlling the motion, managing the power, and so on). A 100-kbit circuitry is expectedly able to manage the information coming from a few (of the order of 10) sensing regions, each specialized to the identification of a different metabolite. Starting from the reasonable assumption that in the next 20 years integrated circuits will attain a density on the scale of 10^{11} cm^{-2} and that the nanobot will be fabricated employing the planar technology, hosting 10^5 devices requires an area of 10^2 μm^2. The nanobots need also to be equipped with sensors, power supply and biomimetic coating (see [6] for more details). Moreover, the nanobot controller has to be very simple; two possibilities are a finite-state machine or a subsumption architecture [2], based on a time-independent behavior arbitration module .

A ellipsoidal disk, with major axis $2a$ of about 50 μm, minor axis $2b$ of 2.5 μm, thickness h of 1 μm and suitably terminated, can move quite freely through the entire circulatory system. Hence the idea of surveying the state of the organism with swarms of such devices in the blood. Such a disk could host 10^5 bits (suitable for hosting the needed circuitry), and could be produced via planar technology, building the device on silicon-on-insulator (SOI) substrate, and removing the substrate at the end of process. To avoid that in the blood stream the nanobot undergoes rotation (that in arterioles, venules and capillaries might be harmful) it must also be endowed with suitably designed hydrodynamic appendages. When immersed in fluid in laminar flows, discs shaped with low aspect ratio are known to undergo lateral drifts toward the vessel wall [9] and this phenomenon can be exploited for their docking thereon. Assuming for a while that such a chip (sketched in Fig. 2) can actually be built, it has shape and size allowing it to explore the whole organism through the circulatory system. To have an idea of the 'invasiveness' of the surveillance system, consider that the total number of white blood cells in an adult human organism is typically $2.0 - 5.5 \times 10^{10}$.

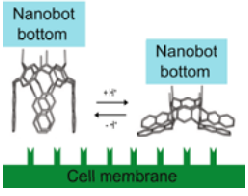


Fig. 3. Exploiting the cave-to-kite change of conformation produced by protonation to allow the docking of nanobots to cell membrane

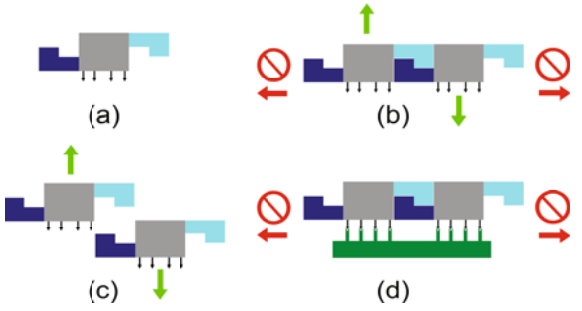


Fig. 4. (a) A nanobot with appendages that allow the adhesion to cell membrane and the formation of clusters. (b) A cluster of two nanobots and the forces acting for its decomposition. (c) The decomposition of the cluster is possible only along selected degrees of freedom. (d) Freezing of the degrees of freedom allowing the decomposition of adsorbed clusters.

Assuming that each nanobot has a mass of the order of 10^{-9} g, the mass of the entire system should be of 3×10^{-4} g.[1]

4 Nanobot Task: Localization and Data Transmission

The first task of the scouts described in the previous section is to locate unhealthy cells, and to report their positions to a central unit outside the body. Due to the limited resources of the nanobots, direct wireless transmission of data to the central unit is, at present, a seemingly insurmountable barrier [6]. This problem could be solved by clustering enough nanobots around unhealthy cells. In fact, if this cluster is sufficiently large, it can trivially be imaged via x-ray computerised axial tomography (CAT). A more sophisticated application exploits the fact that the cluster may become large enough to behave as an antenna, able to send an electromagnetic pulse to the central units — the transmission of radiation in the millimeter band would require the clustering of ~ 10 nanobots only. A related problem (data collection via a swarm) has been discussed in [17]. To accomplish this task, the nanobots need to autonomously disperse in the capillary bed, take chemical sensor reading, mark the region where a positive signal is detected, and form a cluster in that region.

Diffusion. Assuming that the nanobot is transported by the blood at its velocity, the time $\overline{\tau}$ spent by the nanobot flowing along the capillary is around $0.3 - 0.6$ s. We assume that this time is sufficiently long to allow sensing. Since

[1] In this way the total area of the circulating nanobots should be of $0.3 - 0.6$ cm^2, appreciably smaller that of ICs with giga-scale integration (GSI). Assuming the existence of a mature technology for nanobot fabrication, the cost of the entire system should thus be comparable with that of an IC with GSI complexity.

the total time required by blood to make a cycle is about 20 s, each nanobot can explore approximately 1.3×10^5 capillaries per month so that a collection of 3×10^4 nanobots should guarantee that each capillary is on average crossed once per month. Assuming that each capillary must be sensed 10 times per month, the number of the required nanobots is 3×10^5.

Region marking. For marking the region where a positive signal is detected, the nanobot bottom can be terminated with quinoxaline cavitands, as shown in Fig. 3. Due to the hydrophobic character in cave conformation (lhs of Fig. 3), the termination will prevent the docking to epithelial cells. Imagine, however, that once a marker molecule is detected the nanobot produces electrochemically (via electrolysis of water) H^+ ions and inject them at its bottom. Their capture by the quinoxaline nitrogen atoms will cause the opening of the cavity in the kite position (rhs of Fig. 3), due to the Coulomb repulsion strength among positively charged nitrogen atoms [15]. The protonated nitrogen atoms will then be attracted by the negatively charged sites of the cell membrane, forming the glue for the attachment of the nanobot to a nearby cell forming the capillary wall.

Clustering. The most complex subtask is the cluster formation. In swarm robotics, spatial coordination between robots is often critical. When the robots are macroscopic, usually this coordination is achieved via local relative positioning sensors which are based, for example, on ultrasound or infrared technologies. Thus, by using these sensors, nearby robots can communicate and determine the bearing, orientation, and range of their neighbors. In order to maintain a similar communication scheme even between nanobots, Cavalcanti et al. [3] proposed that each nanobot stores specific chemicals to be released for detection by other nanobots. However, another scheme not requiring these chemicals and their related mechanism can be proposed by exploiting a property of the microscopic world: the fact that collisions are not an issue. Of course, this is very different from what happens for macroscopic robots, where collision avoidance is a major issue to be taken into account. *Vice versa*, communication between nanobots can happen through direct physical clashes. In the following the focus is on this last approach which, to the best of our knowledge, has not been investigated yet. For a swarm of 1.3×10^6 agents, a nanobot anchored to a capillary wall will interact with another nanobot each 3 days on average, so that a cluster of 10 nanobots, sufficiently large to inform the central unit, will be formed in approximately one month. A nanobot able to tackle this task has terminations binding selectively to cells and is endowed with two appendages able to bind to one another via weak non covalent forces, and disallowed to do that within the same nanobot by the robust constraints. See Fig. 4 (a). It can be hypothesized that when two nanobots collide in liquid phase (say in blood), they remain paired for a short time. Although this can lead to the random formation of a cluster, it may easily be destroyed by the thermal reservoir, as shown in Fig. 4 (b) and (c). On the other side, clusters may be stabilized when they are formed on the membrane of a cell, as shown in Fig. 4 (d), so that nanobots anchoring on unhealthy cells become themselves sites for docking of other nanobots. This

self-docking mechanism has also the advantage of accelerating the formation of the cluster.

Self-docking. To illustrate this consider a square area of 30×30 (arbitrary units), and that 50 nanobots are introduced in its lower left corner — at coordinate (0,0), see figure 5. Suppose that the target molecule to be identified is in position (20,20). The sensing happens trough direct contact, and we assume that each nanobot can sense a molecule in a range of radius 1. At each time step every nanobot chooses a random direction, and moves 3 units along it. When a nanobot senses the target it stops moving (anchors to its position); now the nanobot becomes itself a target and, since it is much larger than a molecule, it can be sensed in a range of radius 2. Running this simulation 1000 times (with 60 time-steps each) shows that when the anchored nanobots are not used as new target the cluster is on average of 2.78 nanobots, while if they are used as new targets (i.e., new sites for docking) the cluster is on average of 9.91 nanobots. Figure 5 shows the result of one of the simulations. This naive strategy for cluster

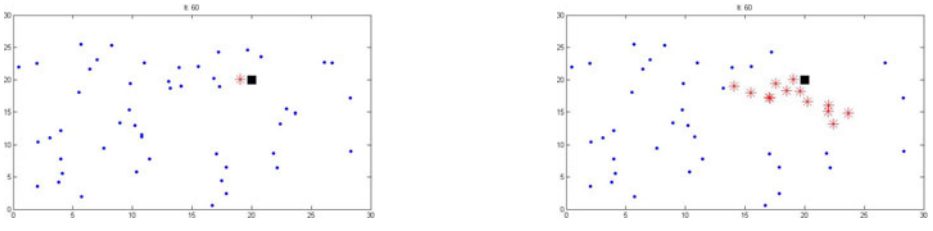


Fig. 5. Cluster (*) of nanobots (·) around the target (■). Left: without self docking. Right: with self-docking.

formation seems to be effective only when the concentration of target molecules is low. In fact, running the same simulation with 10 targets randomly distributed in the area shows that on average only 3 of them are identified, because most of the nanobots tends to cluster around a few targets. This means that to deal with more realistic scenarios more subtle strategies are needed.

5 Conclusions

In this work we described a system, composed of a central unit and a swarm of nanobots, for the early-stage diagnosis of endogenous diseases. A great advantage of nanobots is the fact that they can check the markers of the pathologic tissue in the vicinity of the region where they are generated. On one side, to describe the environment in which the nanobots operate a fractal-tree model of the blood circulatory system has been investigated. On the other side, to identify the position of unhealthy cells it has been proposed to form clusters of nanobots. In particular it has been shown that, for this task, direct physical contact can be a way of communication between nanobots. This mechanism could be used

as basis for developing suitable swarm intelligence algorithms. At last, this work is just the first step of a wider activity. The next steps include the further characterization of the fractal-tree model, and the identification and simulation of suitable swarm intelligence algorithms.

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: from natural to artificial systems. Oxford University Press, Inc., New York (1999)
2. Brooks, R.: New approaches to robotics. Science 253, 1227–1232 (1991)
3. Cavalcanti, A., Hogg, T., Shirinzadeh, B., Liaw, H.: Nanorobot communication techniques: A comprehensive tutorial. In: ICARCV, pp. 1–6. IEEE, Los Alamitos (2006)
4. Cerofolini, G.F.: Two routes to subcellular sensing. In: Korkin, A., Krstic, P., Wells, J. (eds.) Nano and Giga Challenges in Electronics, Photonics and Renewable Energy (NGC 2009). Springer, Berlin (2010) (to be published)
5. Cerofolini, G.F., Amato, P.: Swarms of nanobots for in vivo diagnosis of endogenous diseases (2010), http://asdn.net/asdn/life/nanorobots2.shtml
6. Cerofolini, G.F., Amato, P., Masserini, M., Mauri, G.: A surveillance system for early-stage diagnosis of endogenous diseases by swarms of nanobots. Advanced Science Letters 3 (2010) (to be published)
7. Freitas, R.: Pharmacytes: An ideal vehicle for targeted drug delivery. Journal of Nanoscience and Nanotechnology 6, 2769–2775 (2006)
8. Gabrys, E., Rybaczuk, M., Kedzia, A.: Fractal models of circulatory system. Symmetrical and asymmetrical approach comparison. Chaos, Solitons & Fractals 24(3), 707–715 (2005)
9. Lee, S., Ferrari, M., Decuzzi, P.: Shaping nano-/micro-particles for enhanced vascular interaction in laminar flows. Nanotechnology 20, 495101 (2009)
10. Mandelbrot, B.B.: The Fractal Geometry of Nature. W. H. Freedman and Co., New York (1983)
11. Martel, S., Mohammadi, M.: Using a swarm of self-propelled natural microrobots in the form of flagellated bacteria to perform complex micro-assembly tasks. In: Proc. of the 2010 IEEE Int. Conf. on Robotics and Automation (ICRA). IEEE, Los Alamitos (2010)
12. Murray, C.D.: The physiological principle of minimum work applied to the angle of branching of arteries. J. Gen. Physiol. 9(6), 835–841 (1926)
13. Nagy, Z., Harada, K., Flückiger, M., Susilo, E., Kaliakatsos, I.K., Menciassi, A., Hawkes, E., Abbott, J.J., Dario, P., Nelson, B.J.: Assembling reconfigurable endoluminal surgical systems: Opportunities and challenges. International Journal of Biomechatronics and Biomedical Robotics 1(1), 3–16 (2009)
14. Requicha, A.A.G.: Nanorobots, NEMS, and nanoassembly. Proceedings of the IEEE 91(11), 1922–1933 (2003)
15. Roncucci, P., Pirondini, L., Paderni, G., Massera, C., Dalcanale, E., Azov, V., Diederich, F.: Conformational behavior of pyrazine-bridged and mixed-bridged cavitands: A general model for solvent effects on thermal vase-kite switching. Chem. Eur. J. 12, 4775–4784 (2006)
16. Service, R.F.: Nanotechnology takes aim at cancer. Science 310(5751), 1132–1134 (2005)
17. Winfield, A.F.T.: Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots. In: Parker, L.E., Bekey, G.A., Barhen, J. (eds.) DARS, pp. 273–282. Springer, Heidelberg (2000)

EDA-PSO: A Hybrid Paradigm Combining Estimation of Distribution Algorithms and Particle Swarm Optimization

Endika Bengoetxea[1] and Pedro Larrañaga[2]

[1] Intelligent Systems Group, University of the Basque Country, San Sebastian, Spain
endika@ehu.es
[2] Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Madrid, Spain
pedro.larranaga@fi.upm.es

Abstract. Estimation of Distribution Algorithms (EDAs) is an evolutionary computation optimization paradigm that relies the evolution of each generation on calculating a probabilistic graphical model able to reflect dependencies among variables out of the selected individuals of the population. This showed to be able to improve results with GAs for complex problems.

This paper presents a new hybrid approach combining EDAs and particle swarm optimization, with the aim to take advantage of EDAs capability to learn from the dependencies between variables while profiting particle swarm's optimization ability to keep a sense of "direction" towards the most promising areas of the search space. Experimental results show the validity of this approach with widely known combinatorial optimization problems.

Keywords: Swarm intelligence, Estimation Distribution Algorithms, Particle Swarm Optimization, Bayesian Networks.

1 Introduction and Motivation

Despite the popularity of Genetic Algorithms (GAs) [9] when applied to combinatorial optimization problems their behavior depends largely on the adequate setting of parameters –crossing and mutation operators, probabilities of crossing and mutation, size of the population, rate of generational reproduction...– and GAs show a poor performance in some problems where the designed operators of crossing and mutation do not guarantee that the building block hypothesis is preserved.

Estimation of Distribution Algorithms (EDA) [16,13] were proposed in the aim of making easier to predict the movements of the populations in the search space as well as to avoid the need for so many parameters as in GAs. EDA are population-based search algorithms based on probabilistic modeling of promising solutions. It has been underlined that their higher execution time pays off when optimization problems contain dependencies between variables which EDAs can

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 416–423, 2010.

learn on their learning step and propose the new generation of individuals accordingly [14,15]. Many papers in the literature improve performance focusing on several aspects of EDAs such as adding local optimization to improve each individual, the capability to learn more complex probabilistic graphical models and hybridation of EDAs with other known paradigms.

On the other hand, other evolutionary computation techniques such as Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) focus the search strategy on maintaining a set of entities –ants or particles– which are able to keep a sense of *historical memory*. In the particular case of PSO [10,8], a population of particles is kept moving around the search space at given velocities that are adjusted stochastically according to the historical best position –*gbest*– and the neighborhood best position –*lbest*. If we compare PSO and GAs, PSO have fewer parameters to set and keeps memory capabilities since each particle remembers its best value through *gbest*.

In EDAs this notion of *memory* is not explicitly applied since the newly generated population can be very different from the previous and the probabilistic graphical model learned is based on the characteristics of the selected individuals of the current generation only. This lack of sense of memory in EDAs could be a drawback for concrete optimization problems with local optima having a similar fitness value. Here we propose to combine EDA's capability of learning dependencies between variables together with the memory or directional sense of ACO and PSO, through a new hybrid approach between EDA-PSO.

Hybrid PSO systems are nowadays an active research trend [22]. Examples include hybrids with GAs [7], cooperative approaches [4], self-organizing hierarchical techniques [19], and deflection, stretching and repulsion techniques [18]. Other approaches introduce into PSO techniques inspired in biology to prevent the swarm from crowding too closely and to locate as many optimal solutions as possible [5,17]. In the literature we can also find examples of combining PSO and EDA algorithms, although hybridizing occurs either partially or it is only applied as an improvement of PSO's steps. In [23,21] discrete EDAs are successfully applied to learn a probability model out of the particle current position each generation, in such a way that EDAs are applied to improve PSO in a similar way as GA hybrid variants in [2,7]. Finally, [11] presents a paradigm closer to a full hybrid: For every particle in the swarm two candidate particles are generated, one applying PSO and another with EDA. Every iteration the location to move a particle is determined using the PSO position update equation, and EDA-version particle is calculated by sampling a Gaussian distribution from the kernel in all n dimensions of the problem. The fitness of both particles is calculated, and the the fittest is selected for the next iteration.

Our proposal intends hybridizing beyond the [11] approach by building the new population instantiating the probabilistic graphical model of EDA as well as by maintaining a number P or particles that will create P new individuals each generation. Our approach is presented to continuous domains, which generalises discrete domain approaches. To our knowledge there is no such an approach in the literature combining both paradigms. The outline of the article is as follows:

Section 2 presents EDAs' theoretical background, and Section 3 proposes our hybrid approach EDA-PSO. Section 4 describes the experiments performed and the results obtained, and Section 5 provides conclusions.

2 Estimation of Distribution Algorithms

EDAs [13] are non-deterministic, stochastic heuristic search strategies that form part of the evolutionary computation approaches. The characteristic that most differentiates EDAs from other evolutionary search strategies is that they evolve by estimating the probability distribution of the fittest individuals and then sampling the induced model. EDAs are capable to underlying interdependencies among the encoded variables and to express them explicitly through the joint probability distribution associated with the individuals selected every generation.

More formally, let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a set of random variables, and let x_i be a value of X_i. Then, a probabilistic graphical model for $\boldsymbol{X}$ is a graphical factorization of the joint generalized probability distribution, $\rho(\boldsymbol{X} = \boldsymbol{x})$ (or simply $\rho(\boldsymbol{x})$). The model induced every generation in EDAs is expressed in the form of a directed acyclic graph (DAG) describing conditional interdependencies between the variables on $\boldsymbol{X}$. If $\boldsymbol{Pa}_i$ represents the set of parent variables of variable X_i in the probabilistic graphical model, the factorization of the joint distribution could be written as $\rho(\boldsymbol{x}) = \rho(x_1, \ldots, x_n) = \prod_{i=1}^{n} \rho(x_i \mid \boldsymbol{pa}_i)$.

EDAs are classified depending on the maximum number of dependencies between variables that they consider, typically divided into three categories [13]. The Univariate Marginal Distribution Algorithm for continuous domains (UMDA_c) is a representative example of **univariate** EDAs, where all variables are considered to be independent. In the second category, we have EDAs that can take into account **bivariate** conditional dependencies, where each variable can have a maximum of one parent variable. An example for continuous domains is the greedy algorithm called MIMIC (Mutual Information Maximization for Input Clustering), MIMIC_c. The third category is **multivariate** EDAs, where variables can have multiple parent variables. A representative of this category is EGNA (Estimation of Gaussian Network Algorithm) [12].

3 Hybridation of Estimation of Distribution Algorithms and Particle Swarm Optimization

The novelty of our approach consists on ensuring a full hybridation of both techniques by merging the sub-populations generated by EDA and PSO to take advantage of their respective advantages in optimization. Let consider that P is the number of particles in the search process, and that $R > P$ is the number of individuals in the population, out of which each generation a total of $N \leq R$ individuals will be selected to evolve to the next generation in EDAs. At the same time, the population will be divided in P chunks of same size R/P, and each chunk will be assigned to a different PSO particle for all the search process.

Since the population in EDAs is ordered according to the individuals' fitness values, each particle will represent a different swarm from the fittest to the least fit chunk of individuals. This configuration attempts to be a means to exit from local optima in case of too quick convergence, particularly for problems in which the fitness function is ambiguous (i.e there are different individuals with same fitness) or the fitness difference between not equally promising individuals does not sufficiently reflect it to guide the search process adequately.

The EDA-PSO approach will have the next steps:

1.-Initialization: Generate the first population D_0 of R individuals.
2.-Selection & partition: Select a number N ($N \leq R$) of individuals for EDAs, and partition the population in R/P P chunks. Let's call $chunk_i$ to the i^th chunk, for $i = 1, 2, \cdots P$.
3.-EDA-Learning: Induce the n–dimensional probabilistic model of the N individuals.
4.-EDA-Simulation: Generate D_{l+1}^{EDA} of $R - P$ new individuals by simulating the probability distribution learned in the previous step.
5.-PSO-Position update: For each particle P_i $i_1 \ldots P$, update the $gbest_i$ and $lbest_i$ values using $chunk_i$.
6.-PSO-Create population: Generate the portion D_l^{PSO} of the new population of P new individuals from the new position of each particle.
7.-Fusion of populations: Build the new population $D_{l+1} = D_{l+1}^{EDA} \cup D_{l+1}^{PSO}$.

Steps 2 to 7 are repeated until a stopping condition is verified. The steps to generate the two parts of the new full population D_l are different for D_l^{EDA} and D_l^{PSO}. In the former the classical EDA approach is used, , while the P particles are kept separated ensuring that each is assigned to each chunk, so that all particles represent a different fitness category of individuals' swarm. Under this approach both PSO particles and EDA will take into account the individuals generated by both paradigms for the next generation.

As regards the PSO side, our approach is defined in such a way that when the search process is closer to the optimum the population should be more uniform in each chunk. This approach complements the ongoing research on the provision on PSO for the most adequate inertia weight in [6] and acceleration coefficients to particles during the search [19,22], since in our EDA-PSO algorithm these are regulated by means of the uniformity of the different population chunks.

We formulate this approach using the PSO's canonical version and the classical EDA approach for continuous domains, although the random sampling of uniform distributions is removed since the random component will be provided by the simulation of EDA's probabilistic graphical model every generation. Thus, given the velocity vector $\boldsymbol{V_i} = \left[v_i^1, v_i^2, \ldots v_i^n\right]$ and the position vector $\boldsymbol{POS_i} = \left[POS_i^1, POS_i^2, \ldots POS_i^n\right]$, then each generation the velocity and position of each particle will be updated according to the PSO standard formula.

Also, since EDAs usually include the best individual from D_l also in D_{l+1}, we propose a different *lbest* and *gbest* notion regarding the canonical PSO, such that $\boldsymbol{gbest_i}$ will be the global best of $chunk_i$, and to consider $\boldsymbol{lbest_i}$ as a combination of the current individuals of $chunk_i$.

4 Experiments

Experiments were carried out in order to measure both the behavior and performance of EDA-PSO when applied to classical optimization problems. The selected optimization problems are the ones proposed in [3] to compare evolutionary computation algorithms in continuous domains: Ackley [1], Griewangk [20], and Sphere model. These functions have been chosen due to their very different nature: the Sphere model does not contain any local optimum, Ackley presents several local optima, and Griewangk has still many more local optima although with smaller *valleys*:

1. **Ackley:** The fitness function of this minimization problem is defined as $F(\boldsymbol{x}) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n} \cdot \sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + \exp(1)$. The problem is defined with $-20 \leq x_i \leq 30$, $i = 1, \ldots, n$.
2. **Griewangk:** The fitness function of this minimization problem is defined as $F(\boldsymbol{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$, where the range of all the components of the individual is $-600 \leq x_i \leq 600$, $i = 1, \ldots, n$.
3. **Sphere model:** This is a simple minimization problem defined for $-600 \leq x_i \leq 600$, $i = 1, \ldots, n$, with the fitness function $F(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$.

For all problems the optimum value is 0, which is obtained when the individual has all its variables at 0.

For EDA-PSO the initial population was generated using random generation based on a uniform distribution, and the following combination of settings where tested for the different parameters: EDA type (UMDA$_c$, MIMIC$_c$, EGNA$_{ee}$); Problem size (n= 10, 30, 50); Number of particles (P=5, 10, 30, 50, 100); Population size (R=250); Selection size (N=100, 50). As regards the PSO part, the values set were the standard $\omega = 0.7$, $c1 = 0.1$ and $c2 = 0.2$. In EDA-PSO we choose that the the best individual of D_l is also included in D_{l+1}, so that 249 new individuals are generated per generation. The stopping criterion for all problems was satisfied when the optimum solution was found (assuming this case to be the case when the fittest individual has a fitness smaller than 10^{-6}), or when a maximum of 150 generations were reached.

Each algorithm and combination of parameters was run 20 times for each of the optimization problems, and the main results per algorithm and optimization problem are shown in Table 1. Due to lack of space, we show here the most representative results, which resulted in N=50 for UMDA$_c$ and MIMIC$_c$, and $N = 100$ for EGNA$_{ee}$.

The number of particles shows an important effect in the performance. In the case of Ackley, EDA-PSO results do not improve the standard EDA, but in the case of Griewangk and Sphere results are improved when $P = 10$, although they are worse for $P = 100$. It must be noted that for EGNA$_{ee}$ the values of R and N are too low to allow the EDA to learn adequately the dependencies between variables required to guide search adequately, and the table shows that in PSO contributes to help the algorithm to improve; however, in Ackley its performance is also very dependent on the number of particles P.

Table 1. Mean results of each of the problems and algorithms. Here are cases of $N = 100$ for EGNA_{ee}, and $N = 50$ for UMDA_c and MIMIC_c. The *Gen.* column is the mean generations and *Eval.* corresponds to the mean number of evaluations. In cases in which not all executions converged, *Gen.* shows the convergence percentage and *Eval.* is the mean fitness obtained out of all executions.

	Ackley				Griewangk				Sphere			
	$n = 30$		$n = 50$		$n = 30$		$n = 50$		$n = 30$		$n = 50$	
EDA	Gen.	Eval.	Gen.	Eval	Gen.	Eval	Gen.	Eval	Gen.	Eval	Gen.	Eval
UMDA_c	42.9	86007	57.3	114193	26.1	52424	34.2	68616	24.8	49825	34	68216
MIMIC_c	42.9	86007	57.2	114592	25.9	52024	34.2	68616	24.4	49026	33.6	67416
EGNA_{ee}	70%*	0.72	0%*	2.9	28.8	57821	37.2	74613	27.2	54623	36.2	72614
EDA-PSO	Gen.	Eval.	Gen.	Eval	Gen.	Eval	Gen.	Eval	Gen.	Eval	Gen.	Eval
UMDA_c & $P = 10$	42.8	86683	56.9	115151	25.3	51351	33.2	67301	24	48726	32.8	66493
MIMIC_c & $P = 10$	42.7	86481	56.8	114949	25.4	51553	33	66897	24	48726	32.6	66089
EGNA_{ee} & $P = 10$	90%*	0.6	0%*	1.8	28.2	57205	35.8	72550	26.9	54581	35.8	72550
UMDA_c & $P = 50$	43	90607	57.2	120413	25	52825	32.6	68777	24	49466	32	67518
MIMIC_c & $P = 50$	42.8	90187	61	128389	24.7	52195	32.8	67938	23	48627	32	67518
EGNA_{ee} & $P = 50$	48.7	102571	0%*	1.8	27.7	58492	34.8	73395	26	54924	80%*	-5E-5
UMDA_c & $P = 100$	42.8	94567	57.1	126013	25	55425	32.6	72137	23.6	52346	32	71258
MIMIC_c & $P = 100$	42.6	94127	57.1	126013	25.3	56085	32.8	72577	23.7	52566	32.2	70818
EGNA_{ee} & $P = 100$	90%*	1.4	0%*	0.6	27.7	61362	34.8	76975	26	57624	80%*	-1.2E05

This table also illustrates the complementarity of this hybrid EDA and PSO approach for optimization problems of different complexity. When there are no no local optima like in Sphere, the contribution of our hybrid approach does not show all its potential. For the case of Ackley where local optima have bigger *valleys* than in Griewangk, in all cases an increase of performance is shown with EDA-PSO over EDA, although the choice $P = 50$ appears the best for EGNA while UMDA performs better with $P = 10$. Finally, when having small local optima such as in Griewangk EDA-PSO allows convergence in less generations as well as a small reduction on the number of individuals evaluated to reach converge. In order to check statistical significance between algorithms' behavior the non-parametric tests of Kruskal-Wallis and Mann-Whitney were applied. The null hypothesis of the same distribution densities was tested between EDA and all EDA-PSO executions obtaining $p = 0.706$ for the number of generations and $p < 0.001$ for such of evaluations respectively. On the other hand, considering the cases of EDA and EDA-PSO with $P = 10$ $p < 0.001$ was obtained both for the number of generations and evaluations.

5 Conclusions and Future Work

This work demonstrates the validity of a new full hybrid EDA-PSO approach, analysing its performance depending on the nature and complexity of the optimization problem. Preliminary experimental results to compare the performance of this new approach on typical optimization problems in continuous domains have been shown, and they have been compared with such of continuous EDAs. At the light of the results we can conclude that EDA-PSO has proved to be a new paradigm capable of improving the results of continuous EDAs, where the

number of particles have an important influence on the efficiency of the algorithm and can contribute to better efficiency when set adequately.

There is a lot of space for improvement and future research by applying diverse variants on how to hybridate EDA and PSO when updating each generation $\boldsymbol{lbest_i}$ which we are currently studying. Future work also is to be done with other well known optimisation problems such as Schwefel or Griewangk, and also other problems in which both EDA and PSO are known not to perform efficiently (such as satisfactibility problems).

Acknowledgments. This work has been partially supported by the Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2008- 06815-C02-01 and Consolider Ingenio 2010 - CSD2007- 00018 projects (Spanish Ministry of Science and Innovation) and COMBIOMED network in computational biomedicine (Carlos III Health Institute).

References

1. Ackley, D.H.: A Connectionist Machine for Genetic Hillclimbing. Kluwer, Dordrecht (1987)
2. Angeline, P.: Using selection to improve particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, Anchorage AK, pp. 84–89 (1998)
3. Bengoetxea, E., Miquélez, T., Larrañaga, P., Lozano, J.A.: Experimental results in function optimization with EDAs in continuous domain. In: Larrañaga, P., Lozano, J.A. (eds.) Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation, pp. 181–194. Kluwer Academic Publishers, Dordrecht (2001)
4. van den Bergh, F., Engelbrecht, A.: A cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation 8(3), 225–239 (2004)
5. Brits, R., Engelbrecht, A., van den Bergh, F.: Locating multiple optima using particle swarm optimization. Applied Mathematics and Computation 189(2), 1859–1883 (2007)
6. Chatterjee, A., Siarry, P.: Non linear inertia weight variation for dynamic adaptation in particle swarm optimization. Computers and Operations Research 33(3), 859–871 (2004)
7. Chen, Y., Peng, W., Jian, M.: Particle swarm optimization with recombination and dynamic linkage discovery. IEEE Transactions on Systems, Man, and Cybernetics 37(6), 1460–1470 (2007)
8. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the 6th IEEE Symposium MIcromachines and Human Science (MHS), pp. 39–43 (1995)
9. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE Conference on Neural Networks (ICNN), vol. 4, pp. 1942–1948 (1995)
11. Kulkani, R., Venayagamoorthy, G.: An estimation of distribution improved particle swarm optimization algorithm. In: Proceedings of the Third International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP), pp. 539–544 (2007)

12. Larrañaga, P., Etxeberria, R., Lozano, J.A., Peña, J.M.: Optimization in continuous domains by learning and simulation of Gaussian networks. In: Proceedings of the Workshop in Optimization by Building and Using Probabilistic Models. A Workshop within the 2000 Genetic and Evolutionary Computation Conference, GECCO 2000, Las Vegas, Nevada, USA, pp. 201–204 (2000)
13. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms. In: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2001)
14. Lozano, J., Larrañaga, P., Inza, I., Bengoetxea, E.: Towards a New Evolutionary Computation. In: Advances in Estimation of Distribution Algorithms. Springer, Heidelberg (2006)
15. Miquélez, T., Bengoetxea, E., Mendiburu, A., Larrañaga, P.: Combining Bayesian classifiers and estimation of distribution algorithms for optimization in continuous domains. Connection Science 19(4), 297–319 (2007)
16. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions: I. Binary parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
17. Parrott, D., Li, X.: Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Transactions on Evolutionary Computation 10(4), 440–458 (2006)
18. Parsopoulos, K., Vrahatis, M.: On the computation of all global minimizers through particle swarm optimization. IEEE Transactions on Evolutionary Computation 8(3), 211–224 (2004)
19. Ratnaweera, A., Halgamuge, S., Watson, H.: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on Evolutionary Computation 8(3), 240–255 (2004)
20. Törn, A., Žilinskas, A. (eds.): Global Optimization. LNCS, vol. 350. Springer, Heidelberg (1989)
21. Wang, J., Kuang, Z., Xu, X., Zhou, Y.: Discrete particle swarm optimization based on estimation of distribution for polygonal approximation problems. Expert Systems with Applications 36(5), 9398–9408 (2009)
22. Zhan, Z.H., Zhang, J., Li, Y., Chung, H.H.: Adaptive particle swarm optimization. IEEE Transactions on Systems, Man, and Cybernetics 39(6), 1362–1381 (2009)
23. Zhou, Y., Wang, J., Yin, J.: A discrete estimation of distribution particle swarm optimization for combinatorial optimization problems. In: Proceedings of the Third International Conference on Natural Computation (ICNC), vol. 4, pp. 80–84 (2007)

Emergent Flocking with Low-End Swarm Robots

Christoph Moeslinger, Thomas Schmickl, and Karl Crailsheim

University of Graz, Artificial Life Lab of the Department of Zoology, Graz, Austria
christoph.moeslinger@uni-graz.at

Abstract. This article analyses a flocking algorithm that was developed specifically for small and simple swarm robots. It is similar to traditional flocking algorithms for swarm robots, however it does not need communication nor global information. Its only requirements are at least 4 circumferential distance sensors which can have very limited range. This is possible because our algorithm generates *emergent* alignment of flock members. We show an analysis of our simulations and a short overview of a real robot experiment.

Keywords: swarm robots, emergent behaviour, flocking.

1 Introduction

The phenomenon of flocks, herds and schools is a prime example of emergent behaviour. The elegant movement of flocking birds or a fish school seems highly coordinated, yet it is solely the result of the interactions within the swarm. There have been lots of speculations as to why this complex behaviour might have evolved. Most explanations state that it is advantageous for the individual to be part of a flock to better avoid predators or to increase the foraging success. However, there have been few quantitative investigations of the real animal behaviour because measurements of a moving, 3-dimensional swarm seem almost impossible. Since the proposed advantages of flocking can only be achieved by the swarm and not the individual, flocking can be described as being *swarm-intelligent* [3]. Such swarm-intelligent behaviours are highly interesting and can, for example, be used in the field of *swarm robotics* [5], where a high number of rather simple robots should reach a goal collectively. Flocking algorithms for autonomous agents have been introduced by Craig Reynolds [17] who tried to emulate this behaviour for computer animations. His 'boids' display stunningly natural group movements which are a result of these three simple behaviours:

- *Collision Avoidance* is a basic behaviour for embodied agents.
- *Flock Centering* makes boids stay close to their (nearby) flock mates.
- *Matching of Velocity and Heading* leads to a common direction of movement.

These standard flocking rules also apply to schools of fish and the effects of different attraction and repulsion forces, group sizes and heterogeneity of the groups have been investigated in [18]. Since the introduction of swarm robotics,

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 424–431, 2010.

these behaviours have also been implemented numerous times in both simulations of swarms and real robot swarms. However, all approaches presented for the *Matching of Velocity and Heading* behaviour in real robot swarms included either communication between the robots to exchange positions and headings [11,23,7,1] or (dynamic) leaders [9]. Such communication requires special robotic hardware (e.g. *Bluetooth*, a stable communication channel, a digital compass or even a global positioning system). The implementations of the aforementioned flocking algorithms are not very nature-like because they require communication among the flock neighbors or even the whole swarm. This complexity renders these algorithms unfeasible for a swarm of small and simple robots. The difference to real flocking behaviour is that in real swarms the animals can align because they can identify the heading of each other. Robots usually do not have the possibility to visually derive the heading of their flock mates from the body form unless they use multiple on-board cameras and complex image recognition. Another alignment method in animals is the use of a specialized *lateral line organ* [14] which is present in most fish species, however an emulation of such an organ for robots is very complicated. There are very few communication-less approaches to robot flocking (e.g. [2,4]) who generated a flocking swarm by using an arena with a light beacon and making robots that are illuminated by this beacon behave differently than the robots they cast a shadow on, which results in a common movement towards the light beacon. Although this solution is quite clever, its downside is that it requires a special arena setup. Another communication-less algorithm was introduced by [15] who evolved controllers for a group of 3 minimally equipped robots with a view to generate formation movement. The simulated evolution could indeed produce controllers that allowed 3 robots to engage in different roles (leader or follower) depending on the position in the small 'flock'. However, the evolved controllers only work on a group size of 3 robots and it does not seem like this solution is applicable for bigger swarms.

For us, the requirements of traditional flocking algorithms are in conflict with the concept of swarm robotics where the individual robot is usually small and expected to have very limited abilities [19]. We are interested in working with minimalistic swarm robots which are not capable of long range communication and do not have global information like position and heading. These constraints limit the potential of robot swarms and thus reaching a common goal, like forming an aggregation, is not an easy task. In swarm robotics, such aggregated swarms could be used for collective transport [22] or assembly [21,16]. Therefore it is interesting to research the flocking potential of such minimalistic swarms.

We have shown in [13] that a swarm of such simple robots can flock without communication, which means that the individual robot does not need to know and communicate the exact positions and headings of its neighbours. Instead, our approach is more nature-oriented and relies purely on the sensory perception of the robots. A robot does not need information of all neighboring robots, but only uses the estimated distance of the nearest neighbour in each of its sensor fields. We discretized the robots' sensor fields into different zones (similar to [10]) which either lead to *attraction to* or *repulsion from* other robots. In this paper we will

investigate the sizes of these zones to show that an asymmetric composition can generate *emergent* alignment which negates the need for complex communication or image recognition to achieve alignment and therefore flocking in small robot swarms.

2 Material and Methods

2.1 Algorithm Requirements

Our flocking algorithm has minimal requirements, which means that it can be implemented on simple and therefore very small swarm robots. The algorithm does not require global information about positions or headings, memory, elaborate robot-to-robot recognition or communication. It only needs at least 4 distance sensors with circumferential vision. In swarm robotics, such distance sensors are usually IR-sensors which are used for obstacle detection and collision avoidance. The sensors can be used in *active mode* and *passive mode*. Active mode means that the robot activates the IR-LED at the position of the IR-sensors and checks for reflected light from obstacles. The range of this active IR-sensing is very limited, depending on the LED strength and on the reflecting surface. In passive mode, the robot checks the sensor without emitting light and can thus detect other IR-emitting sources like other robots. This mode of sensing can have a longer range, depending on the light-emitting source. Other distance measuring sensors, like ultrasonic sensors, can be used for our algorithm as well. In preliminary tests we measured the maximum active and passive IR-sensing ranges of two of our real swarm robot types [20,6] to use realistic constraints in our simulations. In our case the maximum sensor ranges were 1 robot-diameter for active obstacle sensing and about 5 robot-diameters for passive robot sensing. This means that in our experiments each robot only has a very limited perception of its surroundings.

2.2 Simulator

We conducted our experiments using a simulator (see Fig. 2B) developed in the multi-agent programmable modeling environment NetLogo [24]. The simulations are mainly used for a *proof-of-concept* and do not incorporate physical properties like sensor or actuator characteristics. What we aim at with this work is to demonstrate the *usability* of our algorithm. Therefore, we simulated a minimalist 4 sensors model (similar to the I-Swarm robot [19]). In our simulations we use a wrapped arena, i.e. there are no boundaries that could hinder the flock. Other simulation parameters are: robot speed (3 robot-diameters per second), sensor measurements (60 per second), sensor errors (5%, random-normal), turn angle (10 degrees), maximum active sensor range (1 robot-diameter), maximum passive sensor range (5 robot-diameters).

2.3 Flocking Algorithm

At first we define what *aggregate* or *flock* means in this paper: If two or more robots are within the passive IR-sensor range of each other they are considered as being *connected*, because they can react on each other. All robots that are *directly* or *indirectly connected* (i.e. there are other connected robots in between) are considered as being part of an aggregate. If the robots in this aggregate are non-randomly aligned and the centre-of-mass of this aggregate moves in a general direction we define that as a flock.

Our algorithm can be represented as a simple finite state machine. All robots periodically emit IR-light from their distance sensors in order to be perceived by other robots. They also poll their active and passive IR-sensors to measure their distance to objects or other robots. In contrast to a camera, IR-sensing has the constraint that a robot can only distinguish one robot per sensor. Since the IR-sensors return the highest value, that means that the robot can only perceive the nearest neighbor in each sensor field. The returned distances are checked against various *thresholds*, depending on the direction of the IR-sensor. Please see [13] for a depiction of the resulting zones. The 3 layers of our algorithm are as follows:

1. First, the sensor in front is polled actively. This is to check if there are close obstacles in front. If the active sensor returns a distance value which means that there is an obstacle in front that is closer than 1 robot-diameter, the robot turns away from that obstacle. This layer leads to basic *collision avoidance*.
2. If there are no objects in the way, the passive sensors in front and at the sides are polled. This is to check if other robots are too close. If the passive sensor returns a low distance value, which means that there is another robot in front that is closer than 1 robot-diameter, the robot turns away from that other robot. This layer is the *flock separation* part of our flocking algorithm.
3. If there are no other robots in close range, the robot polls its passive sensors at the side and rear positions. This is to check if there are other robots around which are too far away. For every sensor that returns a certain distance value, which means that another robot is inside the passive sensor range but too far away, the robot adds up all resulting turns. For example, if another robot is too far away on the left side of the robot and also another robot is too far on the right side of the robot, the robot does not turn. This layer is the *flock cohesion* part of our flocking algorithm.

In the end the robot always moves forward the predefined distance, leading to a continuous movement.

3 Results

3.1 Threshold Analysis

Our simulations investigate the thresholds of the 4 sensor model to test our hypothesis that asymmetric zones can lead to emergent alignment and thus generate flocking. A parameter sweep was performed that changed the threshold

for the attractive zones on the sides and the threshold for the attractive zone to the rear. The minimal threshold for the zones on the sides was 1 robot-diameter because at distances closer than this the robots turn away from each other. The minimal threshold for the zone on the rear was 0 robot-diameters because robots do not have repulsive zones in the rear. The maximum distance for all thresholds was 5 robot-diameters which is the maximum passive sensor range of our simulated robots. The thresholds were changed at 0.25 robot-diameter intervals resulting in $21x17 = 357$ threshold combinations per swarm size (5, 10 and 15 robots). Each robot swarm started aggregated in the middle of an unbounded arena in a starting area whose size was correlated to the swarm size. The robots' positions inside that starting area and their headings were randomized and 50 repetitions for each threshold combination were made. The centre-of-mass (CoM) of the initially aggregated swarm was calculated and its path logged as long as 80% of the whole swarm were part of the flock.

Fig. 1 shows the path length of the CoM of one exemplary robot swarm size and a superimposed average path length for all swarm sizes after 60 simulated seconds. The measured path length depends on the combinations of the side thresholds (x-axis) and rear thresholds (y-axis) and also on the swarm size. The black lines depict identical threshold values for the sides and rear zones. Ideal flocking with instant alignment would result in a maximum CoM distance of 180 robot-diameters. In these simulation runs we also measured the average global alignment of the swarm during 60 seconds by adding the vectors of all flock members each second. When normalized, the results showed almost identical values as the CoM distance measurements, therefore they are not shown in this paper. Generally, side or rear thresholds greater than 4 robot-diameters lead to incoherent swarms which quickly disperse, resulting in minimal CoM path lengths (see dark areas in Fig. 1). Smaller thresholds generally keep the swarm coherent and result in medium CoM path lengths (see dark grey areas in Fig. 1). Only certain threshold combinations where the rear threshold is greater (= further outside) than the threshold to the sides lead to a relatively long CoM path length (see light grey areas in Fig. 1). The influence of swarm size on the flocking ability is also quite visible, but not shown in this paper. The larger the swarm gets, the worse is its mobility. Flocks of 5 robots can move a distance which is up to 50% of the maximum CoM distance with certain threshold combinations (threshold sides: 1.25 robot-diameters; threshold rear: 2.5 robot-diameters). Flocks of 10 robots can move up to 35% of the maximum CoM distance (threshold sides: 1.75 robot-diameters; threshold rear: 3 robot-diameters) and flocks of 15 robots can move up to 29% of the maximum distance (threshold sides: 2 robot-diameters; threshold rear: 3.25 robot-diameters). By superimposing (and averaging) the results from all three swarm sizes we found out that the best threshold combination for small swarms is 2.25 robot-diameters for the side threshold and 3 robot-diameters for the rear threshold.

3.2 Real Experiments

To attest the usability on real robots, we ported the algorithm to 3 e-puck robots [6]. We emulated the minimalist 4-sensors model by combining 2 of the

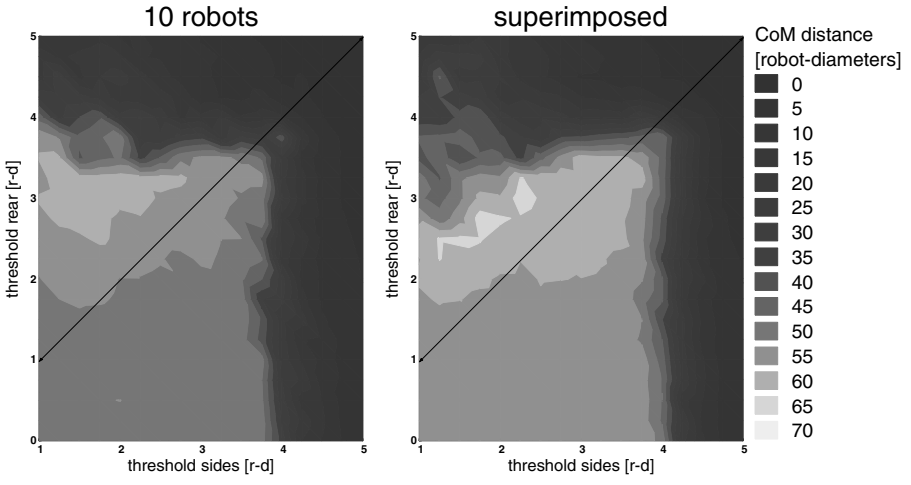


Fig. 1. Results of a parameter sweep which tested the effects of different side and rear threshold combinations on flock coherence and flock mobility. The path length of the centre-of-mass of an initially randomly aggregated swarm is shown as a coloured surface where brighter areas indicate longer path lengths and therefore better flocking of the swarm. Black lines indicate identical threshold values for the side and rear zones. Medians of 50 repetitions for each threshold combination for a swarm size of 10 robots and superimposed results of 5, 10 and 15 robots for 60 seconds.

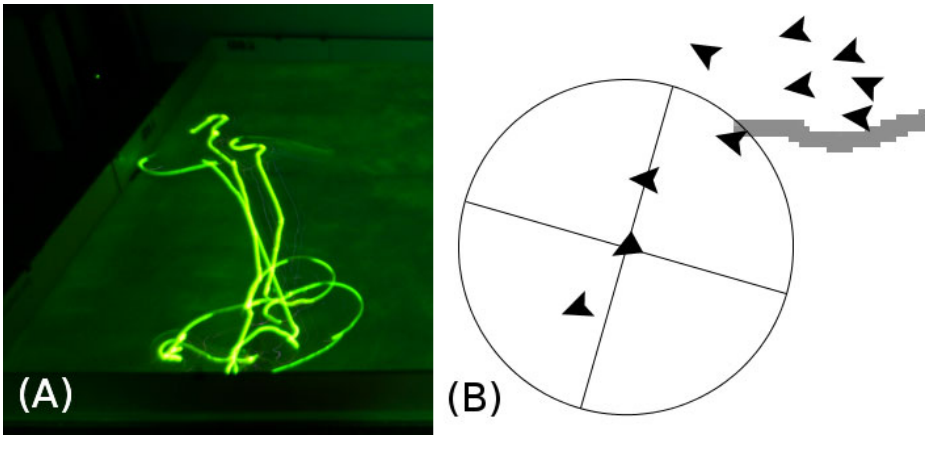


Fig. 2. A: 30 second exposure photo of a test run in a darkened arena in which 3 e-pucks utilised the minimalist flocking algorithm. Each e-puck had 1 green LED, the trails are are visible because of the long exposure. B: Screenshot of a simulated flock of 10 robots (black triangles). The path of the centre-of-mass of the flock is indicated by the grey line on the floor. One of the robots is shown with its sensor range and sensor sectors. A short video of a simulation can be seen at [12].

e-pucks 8 IR-sensors in each direction. Fig. 2A shows a short experiment where the trail of each robot is visible. The small flock managed to stay coherent and also cover a small distance inside the arena. Unfortunately the arena is rather small, so the flock usually reaches a wall very quickly and the arrangement of the flock changes. There are of course a lot of differences to the simulation, mainly sensor characteristics and slower robot speed. Nevertheless we think that our preliminary test runs look promising and we plan on improving the algorithm for the e-pucks and try it with bigger swarms and heterogeneous swarms of e-pucks and Jasmine robots [8].

4 Discussion

In this paper we have analysed a flocking algorithm for swarm robots that works with minimal equipment. This was done by taking a more nature-like approach towards flocking which eliminated the need for communication between the swarm robots. We used the robots' IR-sensors equivalently to a very simple visual perception of an animal and thereby allowed the robots to react on their nearby flock mates. We simulated robots with a minimalist design of 4 distance sensors with a very short passive sensor range of only 5 robot-diameters. Even though flocking algorithms only work if they also have an *alignment* part we did not explicitly implement such a mechanism. In our algorithm, this part is an *emergent* property of the algorithm's 3rd layer. We have shown that if the rear distance threshold (delimiting the attractive zone in the rear) is chosen to be more outward than the side thresholds (delimiting the attractive zones to the sides) this leads to an improved movement of the flock. This improved movement is the effect of the emergent alignment which happens when two robots approach each other. One of these robots will be behind the other robot by chance and due to them sensing each other in different zones the robot behind will turn towards the robot in front. As in natural flocks there are no pre-defined leaders, nevertheless will the robot that is in front 'lead' the robots behind it. If a flock encounters other robots which join the flock, the arrangement of the flock can change instantly and other robots can become the leaders. Our flocking algorithm was also shown to work on real robots. For these experiments we used unmodified, non-communicating e-pucks [6]. One advantage of this flocking algorithm is the adaptability, which means that it can be used on different swarm robot designs. It also allows for heterogeneous robot swarms under the condition that the robots use the same distance-measuring method.

Acknowledgements. This work is supported by the following grants: EU-IST-FET 'SYMBRION', no. 216342; EU-ICT 'REPLICATOR', no. 216240; EU-IST FET 'I-SWARM', no. 507006; FWF (Austrian Science Fund), no. P19478-B16.

References

1. Balch, T., Hybinette, M.: Social potentials for scalable multi-robot formations, vol. 1, pp. 73–80 (2000)
2. Baldassarre, G., Nolfi, S., Parisi, D.: Evolving mobile robots able to display collective behaviors. Artificial Life 9(3), 255–267 (2003)

3. Beni, G., Wang, J.: Swarm intelligence. In: Proc. of the Seventh Annual Meeting of the Robotics Society of Japan, pp. 425–428 (1989)
4. Bjerknes, J.D., Winfield, A., Melhuish, C.: An analysis of emergent taxis in a wireless connected swarm of mobile robots. In: IEEE Swarm Intelligence Symposium, pp. 45–52. IEEE Press, Los Alamitos (2007)
5. Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: A taxonomy for swarm robots. In: Intelligent Robots and Systems 1993, vol. 1, pp. 315–325 (1993)
6. ePuck: e-puck desktop mobile robot - website (2009), http://www.e-puck.org/
7. Hayes, A.T., Dormiani-Tabatabaei, P.: Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots. In: Int. Conf. on Robotics and Automation, pp. 3900–3905 (2002)
8. Jasmine: Swarm robot - project website (2010), http://www.swarmrobot.org/
9. Kelly, I.D., Keating, D.A.: Flocking by the fusion of sonar and active infrared sensors on physical autonomous mobile robots. In: Proc. of the Third Int. Conf. on Mechatronics and Machine Vision in Practice, vol. 1, pp. 1–4 (1996)
10. Kunz, H., Hemelrijk, C.: Artificial fish schools: Collective effects of school size, body size, and body form. Artificial Life 9(3), 237–253 (2003)
11. Matarić, M.J.: Designing emergent behaviors: from local interactions to collective intelligence. In: Proc. of the Second Int. Conf. on From Animals to Animats 2: simulation of adaptive behavior, pp. 432–441 (1993)
12. Moeslinger, C.: Video link (2009), http://zool33.uni-graz.at/artlife/flocking
13. Moeslinger, C., Schmickl, T., Crailsheim, K.: A minimalist flocking algorithm for swarm robots. LNCS. Springer, Heidelberg (2010) (in press)
14. Partridge, B.L., Pitcher, T.J.: The sensory basis of fish schools: relative roles of lateral line and vision. Journal of Comparative Physiology 135(4), 315–325 (1980)
15. Quinn, M.: Evolving controllers for a homogeneous system of physical robots: structured cooperation with minimal sensors. Royal Society of London Transactions Series A 362(1811), 2321–2343 (2003)
16. REPLICATOR: Project website (2010), http://www.replicators.eu
17. Reynolds, C.W.: Flocks, herds, and schools. Computer Graphics 21(4), 25–34 (1987)
18. Romey, W.: Individual differences make a difference in the trajectories of simulated schools of fish. Ecological Modelling 92(1), 65–77 (1996)
19. Seyfried, J., Szymanski, M., Bender, N., Estaña, R., Thiel, M., Wörn, H.: The I-SWARM project: Intelligent small world autonomous robots for micro-manipulation. In: Şahin, E., Spears, W.M. (eds.) Swarm Robotics Workshop: State-of-the-art Survey, pp. 70–83. Springer, Heidelberg (2005)
20. Swarmrobot: Project website (2009), http://www.swarmrobot.org/tiki-index.php
21. SYMBRION: Project website (2010), http://www.symbrion.eu
22. Trianni, V., Groß, R., Labella, T., Şahin, E., Dorigo, M.: Evolving aggregation behaviors in a swarm of robots. In: Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., Kim, J.T. (eds.) ECAL 2003. LNCS (LNAI), vol. 2801, pp. 865–874. Springer, Heidelberg (2003)
23. Turgut, A., Çelikkanat, H., Gökçe, F., Şahin, E.: Self-organized flocking in mobile robot swarms. Swarm Intelligence 2(2), 97–120 (2008)
24. Wilensky, U.: Netlogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL (1999)

Exploiting Loose Horizontal Coupling in Evolutionary Swarm Robotics

Jennifer Owen[1], Susan Stepney[1],
Jonathan Timmis[1,2], and Alan F.T. Winfield[3]

[1] Department of Computer Science, University of York, York, UK
{jowen,susan,jtimmis}@cs.york.ac.uk
[2] Department of Electronics, University of York, York, UK
[3] Faculty of Environment and Technology, U.W.E., Bristol, UK
Alan.Winfield@uwe.ac.uk

Abstract. We describe a theory from Herbert Simon that links the structure of complex systems to increased speed of evolution, and argue the position that this theory can be beneficial to evolutionary swarm robotic research. We propose a way of applying this theory to evolutionary swarm robotic systems by manually designing the robot to robot communication mechanisms and keeping these constant, whilst evolving the rest of the robots' behaviours. This allows for robots to evolve independently of each other without breaking any inter-dependencies that may exist between robots in the swarm. Finally we address potential criticisms of our suggested approach, and outline a course of future research in this area in order to verify our proposal.

1 Introduction

Here we propose a means of speeding up the evolution of swarm robotic systems by considering the role of communication within complex systems.

The main obstacle to swarm robotics research is known as "the design problem". This is the question of which behaviours we should engineer at the robot level to produce a collective emergent behaviour at the swarm level. One proposed solution [10] is to *evolve* the robot's mapping between its sensor inputs and its motor and actuator outputs. With this approach the designer does not have to worry about what particular behaviours or rulesets should be incorporated into the robot; these things are automatically created during artificial evolution. By extending the evolution of a robot's controller to a whole swarm of evolving robots, we get the field of Evolutionary Swarm Robotics (ESR).

One of the main drawbacks of ESR is that measuring the fitness of a particular swarm phenotype is very slow [4]. Measuring this fitness requires that the swarm be run for long enough to build up a clear picture of how well it is functioning in the world. A standard evolutionary algorithm uses tens or hundreds of candidate solutions (here, swarm phenotypes) in the population at each generation, and the algorithm is run for hundreds or thousands of generations [6]; consequently many thousands of fitness function evaluations are made. If each takes several

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 432–439, 2010.

minutes to evaluate, then a whole evolutionary algorithm will take hours to run, which is likely to be longer than the battery life of the robots. Hence, the speed of the fitness function evaluation and of swarm robot evolution in general is a major hurdle for ESR.

2 Complex Systems and Evolution

Simon [9] argues that complexity in systems will lead to faster evolution than in systems without complexity. He is referring to evolution in the Darwinian sense of the gradual change of a species' genome over multiple generations. However, his argument is worded in general terms and can be applied to the formation of non-biological complex systems. He uses an example of atoms forming molecules, which are then used to form amino-acids, and then proteins. The gradual development of structures with low complexity (atoms) to higher complexity (proteins) is also be referred to as "evolution" by Simon. Consequently Simon's argument can apply to both biological and non-biological systems.

2.1 Complex Systems as Hierarchies

Simon [8,9] observes that complex systems have "hierarchic structure". He defines a hierarchic structure as: *"A system that is composed of interrelated subsystems, each of the latter being, in turn, hierarchic in structure until we reach some lowest level of elementary subsystem."*

Take as an example the human body: at the highest level is the body itself; within the body are organs that interact to keep the body alive; each organ is made of interacting cells; cells are made of interacting molecules. In a complex hierarchic structure, at each level the interacting components are subsystems that are complex hierarchies too. Therefore each hierarchic complex system is a system of systems.

Simon [8] further observes that a complex system not only has a hierarchic structure, but also its functionality cannot be understood or recreated from examining its individual subsystems. On each hierarchic complexity level there are interactions between the subsystems and it is from the subsystems and their interactions that we get the higher level complex system. The subsystems themselves are the result of interactions between their own subsystems. From the bottom up, at each successive complexity level, we gain some knowledge of behaviour that was not apparent from observing individuals at the level below. Simon [8,9] calls this "near-decomposability"; we now call it *emergence*.

2.2 Linking Complexity and Evolution

It has been argued that if a system is complex, it will evolve faster than if it is not [5,8,9]. It is certainly true that complex systems are everywhere in nature—take any cellular organism as an example—so there must be some reason that these organisms have prospered whilst non-complex biological systems have not.

Simon [9] examines this relationship between complexity and evolution in depth, and suggests two major reasons why complexity speeds up evolution: stable intermediate subsystems, and loose horizontal coupling.

Stable Intermediate Subsystems. The first observation made in [9], which is also noted in [5,8], is that *"complex systems will evolve from simple systems much more rapidly if there are stable intermediate forms than if there are not"* [8]. Stability in this context means that, despite external perturbation, a system is independently able to maintain some internal state that allows it to continue functioning within its natural environment [7]. In a complex system then, if its subsystems have stability, it means they are able to repair and maintain themselves, and are not as likely to degrade over time. Consequently, stable subsystems are likely to be more prevalent, more available to come together to form something new. Perhaps it is the case that evolution, by developing stable building blocks, speeds the formation of complexity hierarchies, and not the other way around.

Loose Horizontal Coupling. Simon's second observation [9] on how complexity speeds up evolution is to do with functional equivalence and "Loose Horizontal Coupling".

Within a complex system there is "vertical coupling" between levels, in that higher levels are composed of the lower levels. There is also "horizontal coupling": communication and interactions between subsystems on the *same* hierarchic level. If two subsystems on the same level interact with each other in a *fixed* manner, then each is able to evolve independently of the other. These changes may affect the higher level system, and this would direct the evolution. For example, in the human body there is a digestive system and a circulatory system. The digestive system breaks down food and puts it somewhere where it can be absorbed into the bloodstream. If the circulatory system were to route blood more efficiently, then, as long as it still absorbs food from the digestive system, these changes would not affect the functionality of the digestive system. Consequently the higher level system, the body, would be improved.

This relative independence of subsystems due to the fixed nature of their interactions is what Simon calls "Loose Horizontal Coupling" (LHC).

Alphabets. Key to achieving LHC is fixing interaction via a limited "alphabet" of components: *"the flexibility of coupling among subsystems can be further enhanced by limiting the variety of different kinds of components that are incorporated into the larger system"* [9]. Simon uses the example of amino acids. There are 20 types of amino acid, giving an alphabet of size 20 on this complexity level. By repeating and combining parts of this alphabet we can create *"innumerable protein molecules"* [9]. Amino acids are fixed; it is the proteins constructed from them that evolve. An alphabet should be varied enough to be capable of expressing anything, but flexible enough that meaning can be gained from a composite structure of alphabet elements [9]. An alphabet can form the fixed unit of "currency", or information, exchanged between subsystems.

Alphabets are key to LHC because they make it easier for systems to communicate with each other. With an alphabet to dictate what can be communicated, there are fewer types of component to generate in order to pass information between subsystems. There are fewer data types, but the order and structure of the data is what conveys information. In our example of a digestive system with a circulatory system, food is broken down into the message given to the circulatory system. A low complexity level alphabet of amino acids can carry out this communication. A whole range of amino acids are presented by the digestive system; a particular type of protein however, is much more specific and our food is less likely to contain it, so it is less useful for general communication.

In summary, alphabets can regulate the communication between subsystems. They are well suited to this role because alphabet components are more prolific and less specific, compared to things that are composed of the components.

3 Swarm Robots and Speedier Evolution

We have outlined Simon's argument about how complexity in a system can increase the speed of its evolution. A swarm robotic system is complex and dynamic, incorporating feedback and interaction on many hierarchic levels. We conjecture that **the principles of stable hierarchic subsystems with loose horizontal coupling can be applied to evolutionary swarm robotics, resulting in faster evolution of the robot swarm**. The question then arises of how we apply these ideas to swarm robotics.

Addressing the idea of stable subsystems first, the robots in a swarm are themselves stable. Viewing a robot as just a hardware platform for running its controller, the robot will maintain its internal state of being a hardware platform and will continue to do so until either hardware or the battery wears out, or the robot is subjected to destructive external perturbations. The controller within the robot may not be stable, and this could cause the robot to behave erratically. Research has been done into evolutionary algorithms that modularise parts of the genome for reuse, for example [3,2]. Depending on the ability of the chosen evolutionary algorithm to select stable and useful parts of the genome for modularisation and proliferation, stable subsystems within the robot controller could be generated.

Applying loose horizontal coupling (LHC) to robot swarms is more of a challenge because there is so little previous research in this field, particular in the context of evolutionary algorithms and swarm robotics.

Trianni [12] identifies three ways in which swarm robots communicate with each other [12]: *indirect communication* or stigmergy, *direct interaction* where robots physcially interact to communicate and *direct communication* where messages are passed between robots without them needing to physically interact.

We propose that LHC can be implemented by using a fixed means of communication to dictate how the direct communication and interaction between robots should take place. A small fixed alphabet should be used to compose the messages that are conveyed between robots at the relevant subsystem level. This

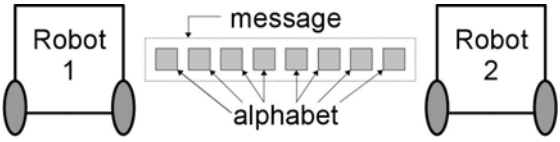


Fig. 1. LHC between two robots. The alphabet and the medium for communication are both designed, and are fixed throughout the evolution of the swarm. Each robot's controller is evolved to map between the robot's sensor inputs (including any received messages) and its outputs. This gives us the robot's behaviour and its interpretation of the message. The message itself emerges from the structure of the combination of several alphabet components and the interpretation that a robot gives to a message.

limits the freedom of the communication, but gives the messages enough flexibility to express meaning in the way the alphabet has been combined. In this manner the functional equivalence of each robot is maintained for as long as that robot can generate and communicate the desired information. If this process is unaffected by the swarm's evolution then LHC will exist between robots, and they can evolve independently of each other without causing other dependant robots to break down. Figure 1 illustrates what we are proposing.

This approach places some restrictions on the freedom of the swarm evolution. LHC is best used when the swarm is evolving in a decentralised way, which is to say, each robot evolves independently with no global time step dictating when to update their genome, using only local information to measure its fitness. LHC in this situation ensures that the robots can evolve independently and still understand each other. If the swarm is evolved centrally, using a global controller to decide robot genomes or fitnesses, (as in [12, 13]) the principles can still be applied but may be of less benefit.

When using LHC, the medium of communication and what alphabet to communicate must be decided *a priori*, so some manual design of the controller is required to support these decisions. This potentially reduces the benefits of evolving the controller as the programmer is still required to design some parts of it. Despite this, it may still be simpler to have to design only the direct communication and interactions, compared to designing the entire controller. Care must therefore be taken when designing the LHC between robots, as the decisions may end up locking the swarm into a behaviour that is less than optimal, and it might never reach maximum fitness. However, over the time frame of a swarm robot experiment we may be able to evolve only over a limited number of generations, so by speeding up the evolution we will hopefully allow the swarm to reach a higher fitness than would have been achievable without LHC. Whilst this may potentially lock us into a lower overall fitness, the benefits should outweigh the costs and we would at least end up with a behaviour that is "good enough".

4 Potential Criticisms

There are some criticisms that might be made when considering our proposal. We address each of them in turn.

The paper from which we are drawing our ideas [9] was written nearly 40 years ago, and so ideas and definitions may be out of date. Simon's hypothesis about how complexity and evolution are linked is based on his understanding of a complex system, described in section 2.1. We have shown that the relevant parts of this definition can be applied to swarm robotics in section 2.2. It therefore follows that Simon's hypothesis can be applied to swarm robotics. Whether his view of complexity is right or not is unimportant. It is the structure and interactions of the complex system that cause the rapid evolution, and we have shown that these are present in swarm robotic systems.

LHC is already implicitly used in evolutionary swarm robotics. Some researchers implicitly use LHC in their experiment. For example, Trianni *et al.* [13] evolve a controller that performs coordinated movement in a swarm of four robots. Before the experiment begins the robots are connected together by the experimenters and they are able to communicate with each other only by exerting a pull on the directly connected robot. The robots are free to evolve the interpretation of this pull, but LHC is implicit in the experimental setup because their means of communication with each other, and the alphabet used to encode the information conveyed, remains fixed throughout the experiment. The alphabet, in this case, consists of pulling forces exerted on the robot in different directions and with different amounts of force.

Similarly, Trianni *et al.* [11] implicitly employ LHC. Each robot in the swarm emits a continuous tone, and the group must evolve their controller to perform swarm aggregation. The continuous tone is used as an "I am here" message communicated between robots, which can be located by other robots using four inbuilt microphones. The robots in this experiment must weight a neural network connecting the microphones and proximity sensors to the motor outputs. In doing so they interpret the signals they receive in order to aggregate together. The direct interaction and direct communication has been explicitly pre-specified by the experimenter.

In experiments where a solitary robot must learn to adapt to a static environment, for example evolving obstacle avoidance behaviour [1], the interaction between the sensed object and the data returned by the robot's proximity sensors is fairly consistent. Hence there is LHC between the robot and its environment because the communication between the two is fixed and does not change over the course of the experiment. In this case the LHC is implicit in the experiment since the experimenter has not specified that the robot's interactions are fixed. LHC is instead just an artefact of how the robot observes its world.

In these examples the LHC is either implicit or unintentional. Although we have not given a very thorough analysis of the field of evolutionary swarm robotics, these examples are sufficient to show that LHC is already implicitly

used in some current evolutionary robotics research. We suggest that the implementation of LHC should be *explicitly* considered when conducting future experiments. This is because it affects how the robots evolve, and helps us to understand from what starting point we are evolving the swarm. We can also assess how easy are we making things for the evolutionary algorithm, so that its effectiveness can be assessed and compared to those of other experiments.

Is lack of LHC even a sensible alternative? Are there any circumstances where not using LHC, even implicitly, makes sense?

In the example of [1], where obstacle avoidance behaviour is evolved, the implicit LHC is due to the static environment causing consistent proximity data. If the world were dynamic, the interaction between proximity sensors and objects in the world would not be so consistent, because objects could be moving towards or away from the robot. Consequently, for there to be no implicit LHC present, the environment must be dynamic. But this is the case with swarm robots, because having multiple agents in the environment causes motion and changes to occur in the environment.

In the case of [11] LHC could be removed by stopping the continuous tone, and leaving the robots to evolve the ability to know when to turn them on or off. In [13] the LHC could be removed by separating the robots and leaving them to connect together themselves, although this would completely change the nature of the experiment, which was to measure whether the robots can learn coordinated movement. Essentially in both cases the LHC is removed by leaving the robot-to-robot communications to be completely evolved, and care must be taken to make sure there is no implicit LHC in the experiment.

Not using LHC in the experiment is sensible if the aim is to evolve the robot swarm from nothing, with no pre-established direct interaction and communication between robots. If evolving all this is not practical, or would fundamentally change the nature of the experiment, then removing LHC completely is not a sensible option. It depends on what goals you want to achieve.

5 Conclusion and Future Work

We have presented Simon's idea of how stable subsystems with loose horizontal coupling increase the speed of evolution [9]. The point we make in this paper is that the presence of LHC and stable subsystems may be used speed up evolution in swarm robotic systems by fixing the swarm's methods of robot interaction and communication. Not to use LHC might be desirable in some scenarios, but if the ideas presented in this paper prove correct and LHC is used, whether implicitly or explicitly, we should be aware of the fact and hence maximise its effectiveness.

Our next stage is to investigate Simon's hypothesis using an evolutionary robotic swarm. We will measure the rate of evolution in the case of a swarm using LHC and compare it to the rate of evolution if the swarm were to not use LHC. If we can show that Simon's hypothesis is effective in speeding up the evolution process in robotic systems then our work will help to present evolution

as a more viable solution to the "design problem" of swarm robotics. It would also help verify that Simon's hypothesis is correct. This would have consequences in the application of evolutionary algorithms more generally, because the same principles could be applied to the evolution of other classes of solutions.

In this paper we have focused on the robot level of the swarm, and how we could implement LHC. We have not yet considered the possibility of using stable subsystems with LHC *within* the robot controller. If we could develop stable modules within the controller and have these maintain their inputs and outputs as part of their stability, then there is greater potential for speeding up swarm evolution. This is, however, a far more complicated task and further research is required in this area.

Acknowledgements. This work is part of the CoSMoS project, funded by EPSRC grants EP/E053505/1 and EP/E049419/1.

References

1. Floreano, D., Mondada, F.: Automatic creation of an autonomous agent: genetic evolution of a neural-network driven robot. In: From animals to animats 3, Brighton, UK, pp. 421–430. MIT Press, Cambridge (1994)
2. Forrest, S., Mitchell, M.: Relative building-block fitness and the building block hypothesis. In: Proc. 2nd Workshop on Foundations of Genetic Algorithms, pp. 109–126. Morgan Kaufmann, San Francisco (1993)
3. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
4. Lund, H.H., Hallam, J.: Evolving sufficient robot controllers. In: 4th IEEE Int. Conf. on Evolutionary Computation, pp. 495–499. IEEE Press, Los Alamitos (1997)
5. Maturana, H.R., Varela, F.J.: Autopoiesis and Cognition: the Realization of the Living. D. Reidel (1980)
6. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge (1996)
7. Pimm, S.L.: The complexity and stability of ecosystems. Nature 307(6), 321–326 (1984)
8. Simon, H.A.: The architecture of complexity. Proceedings of the American Philosophical Society 106(6), 467–482 (1962)
9. Simon, H.A.: The organization of complex systems. In: Pattee, H.H. (ed.) Hierarchy Theory, pp. 1–27. George Braziller (1973)
10. Trianni, V.: Evolutionary Swarm Robotics: Evolving Self-Organising Behaviours in Groups of Autonomous Robots. Springer, Heidelberg (2008)
11. Trianni, V., Groß, R., Labella, T.H., Şahin, E., Dorigo, M.: Evolving aggregation behaviors in a swarm of robots. In: Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., Kim, J.T. (eds.) ECAL 2003. LNCS (LNAI), vol. 2801, pp. 865–874. Springer, Heidelberg (2003)
12. Trianni, V., Labella, T.H., Dorigo, M.: Evolution of direct communication for a swarm-bot performing hole avoidance. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 131–132. Springer, Heidelberg (2004)
13. Trianni, V., Nolfi, S., Dorigo, M.: Hole avoidance: Experiments in coordinated motion on rough terrain. In: Proc. 8th Conference on Intelligent Autonomous Systems (IAS8), pp. 29–36. IOS Press, Amsterdam (2004)

Formal Verification of Probabilistic Swarm Behaviours

Savas Konur, Clare Dixon, and Michael Fisher

Department of Computer Science, University of Liverpool, Liverpool, UK
{Konur,CLDixon,MFisher}@liverpool.ac.uk

Abstract. Robot swarms provide a way for a number of simple robots to work together to carry out a task. While swarms have been found to be adaptable, fault-tolerant and widely applicable, designing individual robot algorithms so as to ensure effective and correct swarm behaviour is very difficult. In order to assess swarm effectiveness, either experiments with real robots or computational simulations of the swarm are usually carried out. However, neither of these involve a deep analysis of *all* possible behaviours. In this paper we will utilise automated *formal verification* techniques, involving an exhaustive mathematical analysis, in order to assess whether our swarms will indeed behave as required.

1 Introduction

A *robot swarm* is a collection of simple, and often identical, robots which will work together to achieve some task [1,2,12]. Whilst the behaviour of each individual robot is fairly easy to understand, it is considerably harder to predict and guarantee the emergent behaviours of the overall swarm. Consequently, it is very difficult to design an individual robot control procedure that, when replicated across all the robots, will guarantee the required swarm behaviour. In this paper we will explore how such robot algorithms can be analysed in a more formal way than simply by simulation or testing.

To exhibit our approach, we have chosen a scenario for which swarm algorithms have already been designed, implemented and tested. Thus, we focus on *foraging robots*, specifically those developed in [10,11]. The idea is that robots have to search for, and retrieve, food items, bring them back to the '*nest*' and then rest. In analysing swarm algorithms of this sort, it is particularly important to see how system parameters affect the swarm behaviours in terms of, for example, *overall swarm energy* or the ratio of *searchers* to *resters* within the swarm. By such an analysis we can explore under what conditions the swarm exhibits optimal behaviour.

Swarm analysis is usually carried out either by testing real robot implementations, or by computational simulations; see, for example, [7,11]. However, each of these only examines a relatively small number of the possible swarm behaviours and so, especially where swarms are to be deployed in safety (or business) critical areas, these approaches guarantee very little about actual swarm behaviours.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 440–447, 2010.

A general alternative to simulation and testing is to use *formal verification*, and particularly the technique called *model-checking* [3]. Here a mathematical model of *all* the possible behaviours of the system is constructed and then all executions through this model are assessed against a logical formula representing a required property of the system. If there is any possible behaviour that violates the required property, then this is highlighted.

While model checking techniques usually assess a *temporal* property, we will use something more sophisticated. Since there is inherent uncertainty within swarms, here we use a *probabilistic* model checker called PRISM [6]. Thus, we use a *probabilistic model* of the individual robots [10] and so we can potentially analyse not just the temporal, but also the probabilistic, properties of the swarm. So, our aim is to take an existing robot swarm algorithm, model it in our framework, and then automatically analyse all possible runs through the system via the PRISM model checker. While we can verify properties of individual robots, we are particularly concerned with *global* swarm behaviour. Thus, we will show that we can not only re-create the simulation results from relevant papers [10], but can also show that certain properties hold of *all* possible runs/configurations, i.e. we can *formally verify* swarm behaviour. This extends beyond testing and simulation of robot swarms to formal verification via probabilistic model checking and provides the potential for much deeper analysis of swarm behaviours.

In Section 2 we introduce the foraging robot scenario and in Section 3 we explain how it is to be modelled and verified. In Section 4 we present several experiments based on this model. These are carried out using the PRISM model checker and comprise both simulation and verification activities. In Section 5, we provide concluding remarks.

2 The Foraging Robot Scenario

The foraging robot scenario we consider follows that presented in [10]. Within a fixed size arena, there are a number of foraging robots, i.e. they must search a finite area and bring food items back to the nest. Food is placed randomly over the arena and more may appear over time. The food items collected will increase the energy of swarm, but searching for food items will use up energy and there is no guarantee that robots will actually *find* any food. The behaviour of each robot in the system is represented by the probabilistic state machine in Fig. 1, with states: SEARCHING, where the robot is searching for food items; GRABBING, where the robot attempts to grab a food item it has found; DEPOSITING, where the robot moves home with the food item; HOMING, where the robot moves home without having found food; and RESTING, where the robot rests for a particular time interval. Associated with these states are both time-out conditions and probability values: T_s: the maximum amount a time a robot can continue searching; T_g: the maximum amount of time a robot can attempt grabbing; T_d: (T_h/T_r) the average time spent depositing (homing/resting); γ_f: the probability of finding a food item; γ_g: the probability of grabbing a food item; and γ_l: the probability of 'losing' sight of a food item, e.g. due to robot interference. The

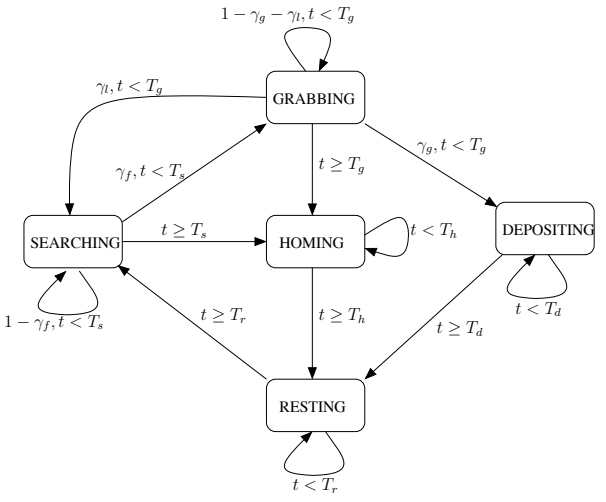


Fig. 1. Probabilistic Finite State Machine for a Single Foraging Robot [10]

probabilistic finite state machine in Fig. 1 is adapted from [10], the main difference being that we ignore avoidance in this initial investigation. All robots are initially in the state SEARCHING. In each time step, robots move to the GRABBING state with probability γ_f (the chance of a robot finding food). Robots stay in the SEARCHING state with probability $1-\gamma_f$. If a robot cannot find food within T_s time steps, it will move to the HOMING state. It is similar for the other states and transitions.

3 Modelling and Verifying the Scenario

The essence of *formal verification* is to analyse a logical requirement (typically within *formal logic*) against all possible behaviours of the system in question. The fastest, and most widely used, approach is called *model-checking* [3]. Here, a structure (such as a finite-state automaton) describing all possible system behaviours is exhaustively (and automatically) checked against the required logical properties. Thus, given that the state-machine in Fig. 1 is a *probabilistic* model, we can analyse both *probabilistic* and *temporal* properties of such robots using a *probabilistic model checker*, such as PRISM [6]. Indeed, when we verify properties of an individual probabilistic state-machine, our input to PRISM is a probabilistic model (technically a *discrete-time Markov Chain*) and a property which can be represented in a number of *probabilistic temporal logics*, such as PCTL [5]. PCTL can be used to represent quantities such as "the probability a robot eventually reaches the nest", "the probability that the energy in the system is greater than E", etc., as well as standard temporal properties. Using PRISM, we can also compute the minimum or maximum probability over a

range of possible configurations or parameters of a model, producing a form of best/worst-case analysis.

However, in this paper we are particularly concerned with verifying the properties of swarms comprising large numbers of robots. Although PRISM is generally quite efficient, allowing us to analyse models with as many as 10^{10} states (see, for example, [4]), a naive application of PRISM to robot swarm verification may generate many more states. For example, if we built a product state-machine from multiple copies of the Fig. 1 state-machine then the size of the resulting model would be *huge*. So, rather than representing each robot as a different probabilistic state machine and then taking the *product* of all these machines to generate the whole system, we use a *counting abstraction* approach. This is particularly useful if there are many identical, independent processes, as is the case in a robot swarm, and allows us to abstract away from low-level probabilistic details and so just consider global population behaviour.

The basic idea is as follows. Since we know that all the robots are modelled by identical probabilistic state machines, then we actually model the whole system by *one* state machine with exactly the SEARCHING, HOMING, etc, states we saw in Fig. 1. However, to each of these states we add a counter which is used to record how many robots are *actually* in that state at that moment. Thus, if 20 robots are searching then the counter in the SEARCHING state will be 20. By examining Fig. 1 we can work out how many of these should move to GRABBING, how many should move to HOMING, and how many should remain in SEARCHING at each step. Thus, in addition to the 5 states, each state is labelled with a set of difference equations explaining how the numbers of robots associated with each state evolve. It is important to note that we are abstracting away from local probabilities and now considering a more global view.

Due to lack of space we do not show the (difference) equations defining how the numbers of robots in each of the five key states changes over time. But we remark that fractional values of numbers of robots are rounded to the nearest whole number since we work with a discrete state space.

4 Experiments

In this section, we present the results from running PRISM on our model with different properties and parameters. We run them both in *simulation mode*, whereby we generate a random run, and in *verification mode*, whereby we assess all possible runs against a PCTL formula. The properties we check use PCTL syntax, which comprises the usual operators from classical logic such as $\wedge$ (and), $\vee$ (or) and $\Rightarrow$ (implies), as well as probabilities for example $P^{=1}$ (i.e. with probability 1) and temporal operators such as $\Box\varphi$ (φ holds at all present/future moments) $\Diamond\varphi$ (φ holds at some present/future moment), $\varphi\,\mathcal{U}\,\psi$ (φ holds until ψ holds).

Energy Calculation. Before discussing the experiments, we will explain how the swarm energy is calculated. In a swarm, each robot consumes a certain amount of energy at each time step. We assume that a robot consumes E_s, E_g, E_r

and E_h units of energy at each step in SEARCHING, GRABBING, RESTING and HOMING, respectively, and each food-item delivers the swarm E_d units of energy (we assume that E_d is net energy, i.e. it is the energy obtained from the food carried minus the energy consumed in the depositing state; we also assume that a robot can carry only one food-item.) The total swarm energy in the next time instance, denoted by $En(t+1)$, is calculated as follows:

$$En(t+1) = En(t) + E_d N_d^{T_d}(t) - E_s N_s(t) - E_g N_g(t) - E_r N_r(t) - E_h N_h(t)$$

where $N_d^{T_d}(t)$ denotes the number of robots that have been in DEPOSITING for T_d time steps; $N_s(t), N_g(t), N_r(t)$ and $N_h(t)$ denotes the number of robots that are in the SEARCHING, GRABBING, RESTING and HOMING states, respectively, at time t. It is important to note that in order to ensure that we have finitely many states we discretise the energy values, and model En as discrete state variable.

4.1 Swarm Model with Resting Timeout

We will start our experiments with a model based on the state structure in Fig. 1 and then will gradually enhance this to be more sophisticated. So, we begin with experiments based on the basic model in which robots leave the RESTING state after waiting for T_r time steps. The energy parameters used are as follows: $E_d = 250$, $E_r = 2$, $E_h = 4$, $E_s = E_g = 50$, and $En(0) = 2000 \times 10^3$ (the initial swarm energy at $t = 0$). We also take $T_s = T_g = T_h = T_d = T_r = 50$ seconds, and $\gamma_l = 0.1$, $\gamma_g = 0.8$. In the sequel, we take N as the total number of robots. Fig. 2 compares the total energy of a swarm of 100 robots with the different (but constant) γ_f (probability of finding food) values. As seen in Fig. 2, if the probability of finding food is below 0.5, i.e. less food is available for the swarm, then the total energy of the swarm decreases. If the probability of finding food is greater than 0.5, i.e. more food is available, then the swarm gains energy.

In PRISM, we can test various runs (i.e. performing simulations as in Fig. 2); but this does not guarantee that a property is true in *all* cases. Using the verification module of PRISM we can also *verify* whether a property holds for all possible runs; or we can determine the actual probability of a property being true. Assume we want to determine the probability that "for an arbitrary number of robots and food finding probability the swarm energy is equal to or greater than the initial energy from a time point t_A". This implies that from a time point t_A the total energy of the swarm never goes below the initial energy. We assume γ_f takes discrete steps with increments of 0.01 in the range $[0, .., 1]$, $N = [100, .., 500]$, $t_A = 2000$ and $t_{max} = 10000$. We can specify this property in PCTL as follows: $P^{=?}((t < t_A \vee En \geq E(0)) \mathcal{U}\, t = t_{max})$. We verified this formula using PRISM, and the probability returned is 0.59. This result validates the simulation shown in Fig. 2. In Fig. 2, γ_f values remained constant throughout the experiment. In Fig. 3 we use a variable γ_f value that decreases over time, i.e. modelling the situation when food gradually becomes more scarce. So, γ_f is 1.0 at $t = 0$, and reduces by 0.1 in every 1000 seconds. Running some simulations, we see that the total swarm energy initially increases, since the probability of

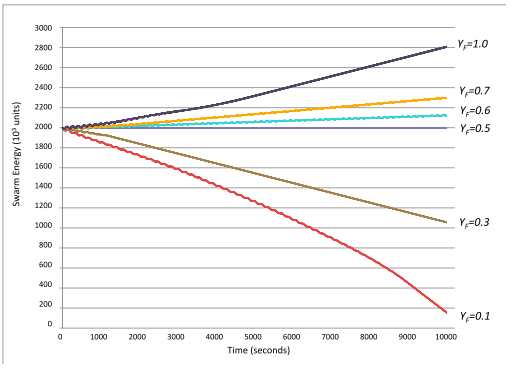


Fig. 2. Total swarm energy vs. the (constant) probability of finding food (γ_f)

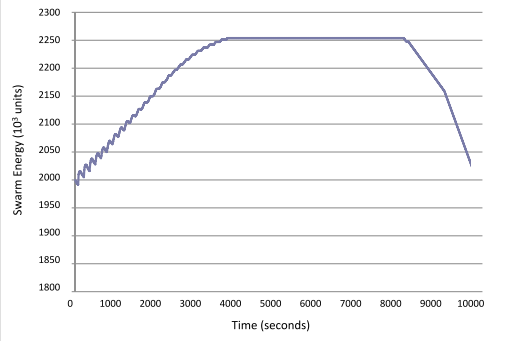


Fig. 3. Total swarm energy vs. the (decreasing) probability of finding food (γ_f)

finding food is high, i.e. there is much food available for the swarm. When the probability of finding food decreases, i.e. food availability reduces, the energy gain becomes equal to the energy spent by the swarm. After a while the energy gained becomes less than the energy spent, since the food becomes scarce, and therefore the total swarm energy decreases.

Again, rather than just running individual experiments, we can *verify* some properties of all executions. For example, the following property states that "the total energy never reduces below a certain value E (before timeout)", specified in PCTL as $P^{=1}\,\square(Energy > E)$. We verified this formula in PRISM (assuming $E = 1900 \times 10^3$ and $N = [100, .., 500]$). This property can be useful to determine a set of robots can achieve an important task within a certain period, since we can be sure they will always have energy above a certain threshold value. We also queried the following formula in PRISM: $P^{=?}(true\ \mathcal{U}\,(En > E \wedge P^{=1}\diamondsuit(En > E \wedge t = t_{max})))$ assessing the probability that if the total energy exceeds E at some time, then the energy level in the end will be above E. PRISM returned a probability value of 0.31 for $E = 3000 \times 10^3$, $N = [100, .., 500]$ and $t_{max} = 10000$.

This is an expected result because food becomes scarce as time passes, and so it is likely that the energy level decreases.

4.2 Swarm Model without Resting Timeout

In Fig. 1, we assumed that if a robot moves to the RESTING state from HOMING or DEPOSITING, it waits T_r time steps in RESTING. We now change this scenario and assume that the robots do not wait in the RESTING state for a fixed amount of time before moving to SEARCHING, but wait there depending on a probability γ_s. This probability, in turn, depends on the number of robots in the DEPOSITING and HOMING states. Namely, robots move from RESTING to SEARCHING states with probability

$$\gamma_s = \frac{\lambda N_d^{T_d}(t) - N_h^{T_h}(t)}{N},$$

and stay in the RESTING state with probability $1 - \gamma_s$. In the above, λ is the *energy gaining parameter*, and $N_d^{T_d}(t)$ (respectively $N_h^{T_h}(t)$) denotes the number of robots that have been in DEPOSITING (respectively HOMING) for T_d (respectively T_h) time steps. Intuitively, we can think of this new scenario as follows: since each robot brings a food item in DEPOSITING state, if more robots move to the DEPOSITING state and less robots move to the HOMING state, then more robots will move to the SEARCHING state. As can be seen, γ_s is proportional to the *energy gaining parameter* λ. That is, if we increase the value of λ, then more robots will move to the SEARCHING state.

Using verification we can also establish several properties. For example, we have checked the PCTL property $P^{=?}\,\Box((t > t_A) \Rightarrow (N_s \geq n))$, specifying the probability that the number of searching robots will never reduce below n after t_A seconds. The probability that PRISM returned is 0.99 for $t_A = 2000$ and $n = 10$. We also assumed λ takes discrete steps with increments of 0.01 in the range $[0, .., 1.5]$ and $N = [100, .., 200]$. This result shows that the number of searching robots is never below n for *almost* all time points from t_A.

In addition to the scenarios above, we also considered a scenario where γ_g depends on the number of robots searching. Thus, if a robot is in the GRABBING state, then it can grab a food with a probability $\gamma_g = \alpha/N_s$, where α is a constant and N_s is number of robots searching. Due to lack of space, we omit the formal analysis of this scenario.

5 Conclusions

In this paper we have taken a probabilistic state transition system for foraging swarm robots from [10] and used it as the basis for verification of global swarm behaviour using the PRISM model-checker. Rather than instantiating such a transition system for each robot and performing local verification, we adopt a macroscopic approach and represent the whole swarm using one transition system calculating the number of robots in each state based on a combination of the number of robots in the previous state and the probability that robots change

state. This allows us to simulate and verify properties of the global foraging robot scenario for a number of parameters. In particular we investigate the changes to swarm energy relating to changing the probability of finding food and differing resting timeouts. We also experimented with variable probabilities including the probability of moving from resting to searching and the probability of grabbing. Using this approach we can formally verify that certain behaviours will *always* happen which cannot be done easily with either simulation or testing.

Foraging robots have been studied in a number of other papers, for example [8,7,9,10,11]. The foraging robot scenario we use here is from [10] however the main difference being that we ignore avoidance in this paper.

Acknowledgements. The authors would like to acknowledge the helpful comments from the reviewers. This work was partially supported by EPSRC research project EP/F033567.

References

1. Beni, G.: From Swarm Intelligence to Swarm Robotics. In: Şahin, E., Spears, W.M. (eds.) SAB 2004. LNCS, vol. 3342, pp. 1–9. Springer, Heidelberg (2005)
2. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. J. Artificial Societies and Social Simulation 4(1) (2001)
3. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (1999)
4. Duflot, M., Kwiatkowska, M., Norman, G., Parker, D.: A Formal Analysis of Bluetooth Device Discovery. Int. J. Software Tools Techn. Transfer 8(6), 621–632 (2006)
5. Hansson, H., Jonsson, B.: A Logic for Reasoning about Time and Reliability. Formal Aspects of Computing 6, 102–111 (1994)
6. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: A Tool for Automatic Verification of Probabilistic Systems. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 441–444. Springer, Heidelberg (2006)
7. Labella, T.H., Dorigo, M., Deneubourg, J.L.: Efficiency and Task Allocation in Prey Retrieval. In: Ijspeert, A.J., Murata, M., Wakamiya, N. (eds.) BioADIT 2004. LNCS, vol. 3141, pp. 274–289. Springer, Heidelberg (2004)
8. Lerman, K., Galstyan, A.: Mathematical Model of Foraging in a Group of Robots: Effect of Interference. Autonomous Robots 13(2), 127–141 (2002)
9. Lerman, K., Martinoli, A., Galstyan, A.: A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems. In: Şahin, E., Spears, W.M. (eds.) SAB 2004. LNCS, vol. 3342, pp. 143–152. Springer, Heidelberg (2005)
10. Liu, W., Winfield, A., Sa, J.: Modelling Swarm Robotic Systems: A Study in Collective Foraging. In: Proc. Towards Autonomous Robotic Systems (TAROS), pp. 25–32 (2007)
11. Liu, W., Winfield, A., Sa, J., Chen, J., Dou, L.: Strategies for Energy Optimisation in a Swarm of Foraging Robots. In: Şahin, E., Spears, W.M., Winfield, A.F.T. (eds.) SAB 2006 Ws 2007. LNCS, vol. 4433, pp. 14–26. Springer, Heidelberg (2007)
12. Sahin, E., Winfield, A.F.T.: Special Issue on Swarm Robotics. Swarm Intelligence 2(2-4), 69–72 (2008)

Inverse Modeling in Geoenvironmental Engineering Using a Novel Particle Swarm Optimization Algorithm

Tadikonda Venkata Bharat and Jitendra Sharma

Department of Civil and Geological Engineering
University of Saskatchewan, Saskatoon, Canada
tvbharat@gmail.com

Abstract. Algorithms derived by mimicking the nature are extremely useful for solving many real world problems in different engineering disciplines. Particle swarm optimization (PSO) especially has been greatly acknowledged for its simplicity and efficiency in obtaining good solutions for complex problems. However, premature convergence of the standard PSO and many of its variants is a downside particularly for its application to the inverse problems. This aspect encourages further research in developing efficient algorithms for such problems. In this work, a novel PSO algorithm is proposed by introducing fitness of a new location in the search space into the standard PSO which enables to enhance the success rate of the algorithm. The proposed algorithm uses center of mass of the population to compare the fitness of global best particle in each iteration. The proposed algorithm is applied to solve contaminant transport inverse problem. The performance of different PSO algorithms is compared on synthetic test data and it is shown that the proposed algorithm outperforms its counterparts. Further, accurate design parameters are estimated using the proposed inverse model from the experimental data.

Keywords: particle swarm optimization, inverse model, contaminant transport.

1 Introduction

Chemical contaminants are the main sources of soil and groundwater pollution. To minimize such contamination, various barrier systems are engineered. As a result, the importance of contaminant transport studies through various barrier materials is widely recognized for the design of landfills and verticle barrier systems. Design of these systems require an accurate estimation of the transport parameters through these containment systems. These design parameters are estimated from the laboratory diffusion tests. The parameters are predicted by matching the theoretical concentration profile with the experimentally observed data of temporal or spatial variation of the concentration. In Geoenvironmental engineering, this is commonly done using visual calibration techniques such as

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 448–455, 2010.

Pollute [11]. A few studies also used the gradient-based optimization techniques for solving this inverse problem [1]. The shortcomings of using the above techniques for contaminant transport problems have been well documented in [4]. Recently, the PSO algorithm has been used for solving the aforementioned inverse problem [4] and for reconstructing the past contaminant source history [3]. Although PSO based solvers predict good solutions, often times they converge prematurely to local solutions. Especially for the application to contaminant transport problems, this is a major set-back as this causes not only improper estimation of design parameters but also wastage of a huge computation (each fitness calculation involves an expensive time marching numerical computations). With this incentive, in this paper a new variant of PSO is presented to improve the efficiency and success rate of the solver for parameter estimation. The proposed version of PSO uses a new location in every iteration inspired from the Big-Bang Big-Crunch (BBBC) optimization concept [6]. The modified PSO algorithm uses fitness of center of mass of the swarm population in addition to fitness of individual. The detrimental position update formula of global best particle in the standard PSO can be alleviated using this additional information in the search space as the center of mass of the population changes randomly in each iteration. This improves the population diversity and enables the swarm to escape from the local optima.

2 Contaminant Transport Problem

2.1 Mathematical Formulation

Consider an engineered barrier system with contaminant cells (landfill), liner system and the leachate collection system. The concentration at any time instant in the contaminant cells (at the upper boundary) can be represented as

$$c(x=0,t) = c_0 + \frac{nD}{H_s}\int_0^t \left(\frac{\partial c}{\partial x}\right)\Bigg|\, x=0 \; dt \tag{1}$$

where H_s is the equivalent height of source reservoir, calculated as the volume of source solution divided by the cross-sectional area of the liner sample perpendicular to the direction of diffusion, D is the effective diffusion coefficient, c is the concentration of the solute in pore fluid at time t, spatial location x, n is the soil porosity and c_0 is the initial contaminant concentration. Similarly, the concentration at any time instant at the lower boundary (leachate collection layer) can be represented as:

$$c(x=L,t) = -\frac{nD}{H_c}\int_0^t \left(\frac{\partial c}{\partial x}\right)\Bigg|\, x=L \; dt \tag{2}$$

where H_c is the equivalent height of the collector reservoir. The one-dimensional governing diffusion equation through liner can be expressed as

$$\frac{\partial c}{\partial t} = \frac{nD}{\alpha}\frac{\partial^2 c}{\partial x^2} \tag{3}$$

where α is the capacity factor represents capacity of the liner material to sorb the contaminant. The initial condition in general encountered is

$$c(0 < x < L, t = 0) = 0 \tag{4}$$

$$c(x = 0, t = 0) = c_0 \tag{5}$$

$$c(x = L, t = 0) = 0 \tag{6}$$

Solute concentration at any spatial location and time instant $c(x, t)$ can be obtained by solving the (2) through (6) simultaneously.

2.2 Numerical Solution

Crank-Nicolson (C-N) numerical solution is used for the quick estimation of theoretical concentrations at any given time step. The discretization procedure of the aforementioned problem using C-N scheme is presented elsewhere [4]. The numerical solution of the forward model estimates, the theoretical contaminant concentration when the design transport parameters are known. However, the laboratory contaminants transport experiments and the field monitoring systems in the landfills yield the spatial or temporal concentration of contaminants. The design transport parameters thus need to be determined from these observations by inverse analysis. In the inverse analysis, rmse (7) is used as an objective function for finding the best set of parameters.

$$rmse = \sqrt{\sum_{i=1}^{L} \frac{(c_{in,anal}(t_i) - c_{in,num}(t_i; D_e, \alpha))^2 + (c_{out,anal}(t_i) - c_{out,num}(t_i; D_e, \alpha))^2}{2n_L}} \tag{7}$$

where $c_{in,anal}$ and $c_{out,anal}$ are the contaminant concentrations data obtained from analytical solution and $c_{in,num}$ and $c_{out,num}$ are the concentrations data obtained by numerical model at given values of t_i, $i = 1, 2, \ldots, N$ theoretical time. The forward model (simple explicit numerical procedure) was integrated with standard PSO for the estimation of design parameters earlier [4]. However, the success rate of the solver is not striking due to premature convergence of the standard PSO algorithm. The detailed description of PSO algorithm is presented in the next section.

3 PSO Algorithms

3.1 PSO Description

PSO is a class of derivative-free, population-based computational method which was developed based on the social behaviors of animals [9]. In this method each agent representing a potential solution moves in the search space and adaptively updates its velocity and position according to its own flying experience and the

flying experience of its neighbors, aiming at a better position for itself. The position of i^{th} particle and j^{th} dimension is represented as $\overrightarrow{x_{ij}} = (x_{i1}, x_{i2}, \ldots, x_{iD})$. The best position of the i^{th} particle in its history that gives the best fitness value is represented as $\overrightarrow{p_{ij}} = (p_{i1}, p_{i2}, \ldots, p_{iD})$. The best particle among all the particles in the whole population and in the entire history is represented by $\overrightarrow{p_{gj}} = (p_{g1}, p_{i2}, \ldots, p_{gD})$. At each iteration step, the position vector of the i^{th} particle x_{ij} is updated by adding an increment vector called velocity v_{ij} as shown below

$$\overrightarrow{x_{ij}} = \overrightarrow{x_{ij}} + \overrightarrow{v_{ij}} \tag{8}$$

The velocity of each individual is updated with the best positions acquired for all individuals over generations, and the best positions acquired by the respective individuals over generations. Updating is executed by the following formula.

$$\overrightarrow{v_{ij}} = \chi\left(\omega\overrightarrow{v_{ij}} + \phi_1 rand_1()(\overrightarrow{p_{ij}} - \overrightarrow{x_{ij}}) + \phi_2 rand_2()(\overrightarrow{p_{gj}} - \overrightarrow{x_{ij}})\right). \tag{9}$$

where χ is called constriction coefficient and ω is the inertia weight introduced by Shi and Eberhart [12] in order to improve the performance of the particle swarm optimizer. ϕ_1 and ϕ_2 are two positive values called acceleration constants. $rand_1()$ and $rand_2()$ are two independent random numbers that uniformly distribute between 0 and 1 and are used to stochastically vary the relative pull of $\overrightarrow{p_{ij}}$ and $\overrightarrow{p_{gj}}$. The positions of respective individuals are updated by every generation, and are expressed by (8).

Shortcomings of Standard PSO Algorithm and Its Variants. A large number of theoretical studies [5] have shown that often times PSO converges prematurely. Many of the particles waste computational effort in seeking to move in the same direction (towards the local optimum already discovered), whereas better results may be obtained if various particles explore other possible search directions [10]. A number of population diversity mechanisms thus have been proposed to improve the performance. The perturbed PSO algorithms are more thriving to overcome premature convergence and successfully applied to solve inverse problems [3]. However, the performance of these algorithms is highly dependent on the perturbation coefficient which varies from problem to problem [2]. Thus finding an optimum value of perturbation coefficient for a given problem will not be practical in many situations.

3.2 Big-Bang Big-Crunch Algorithm

The Big Bang-Big Crunch (BBBC) optimization method [6] is built on, Big-Bang phase: where candidate solutions are randomly distributed over the search space and the Big-Crunch phase: where a contraction procedure calculates a center of mass for the population. A detailed description of BBBC algorithm is given elsewhere [6]. The advantage with this algorithm is that, it uses an additional point on the search space based on the center of mass of the population. However, this algorithm lacks exploration capabilities because of a weak position update formula. Thus, a hybrid algorithm was proposed which uses the personal best

and global best positions of the agents to update the positions of the agents [8]. However, due to exclusion of velocity term in the hybrid BBBC (HBBBC) algorithm, it does not tap the full advantage of original PSO algorithm offered. Thus, a new algorithm is presented which considers the centre of mass concept of BBBC algorithm and position formula of standard PSO algorithm.

3.3 Proposed Algorithm

The pseudo-code of the proposed PSO algorithm can be summarized as follows:

1: Initialize the population by randomly distributing the agents on the search space
2: Calculate fitness of all agents (7)
3: Update the position of each agent using (8) and (9)
4: The path $[s, n_k]$ is a feasible path and destination d_n can be reached
5: Find the center of mass of the whole population using the following equation,

$$x_j^k(c) = \frac{\sum_i^N \frac{1}{fit_i} x_{ij}^k}{\sum_i^N \frac{1}{fit_i}} \quad j = 1, 2$$

where x_{ij}^k is the j^{th} component of the i^{th} solution generated in k^{th} generation and N is the population size.
6: Compute the fitness of the center of mass point
7: Compare the fitness of global best agent and center of mass point. Update the global best value using the greedy selection mechanism
8: Return to Step 3.

4 Inverse Analysis

The goal of the present inverse model is to estimate a combination of mass transport parameters that minimizes rmse (7)) between the experimentally measured and theoretically computed concentration data.

4.1 Parameter Setting

PSO algorithms, in the developed solvers, are based on a population of 25 particles randomly distributed in the solution space. The maximum number of generations is set to 200 in each run. The best set of PSO variables is found by empirical studies and used. It is observed that $\phi_1 = \phi_2 = 0.5$ is found to give good performance when $\chi = 0.6$ and a linearly varying inertia weight ω from 0.9 to 0.4 from the beginning to end of the search.

4.2 Performance Assessment of Different Solvers

To validate performance of different inverse models based on SPSO, HBBBC and PPSO algorithms, a synthetically generated test data is used. The test data of contaminant concentration is obtained by solving the forward problem with the assumed set of design parameters $D = 1.523 \times 10^{-5}$ and $\alpha = 34$. The test data with and without random noise is given as input to the solvers for finding the true design parameters. The performance of all the developed solvers on synthetic data is tested using 10 independent runs. Table 4.2 presents best values, worst values, tolerance and success ratio obtained for each solver. The results indicate that HBBBC and PPSO models achieve better success rate when compared with SPSO model. Though the performance of HBBBC and PPSO models is the same on synthetic data without noise, the superiority of PPSO model is exhibited when the noise is introduced to the data. Thus, it appears that the performance of the PPSO model is not influenced by the noise in the data which is expected in the experiments. Thus, this model is more reliable in estimating the accurate design parameters.

The movements of the particles for HBBBC and PPSO inverse models are analyzed. In case of HBBBC model, the particles have quickly converged (in 100 iterations) to near global optimum solution ($D = 1.168 \times 10^{-5}$ and $\alpha = 45.063$) with a success ratio of 0.5. However, the PPSO algorithm continues to search till the maximum iterations of 200 and converged to global optimum solution ($D = 1.43 \times 10^{-5}$ and $\alpha = 34.805$) with a success ratio of 0.7. It is believed that due to the additional point (center of mass), the poptlation diversity has improved. Further, due to the velocity term, the particles do not change their directions abruptly and thus the proposed method is superior to HBBBC. In order to determine the statistical significance of differences in the mean best values by the HBBC and PPSO models, t-test is performed on test data with

Table 1. Performance of different solvers on synthetic test data

Algorithm	Data Type	Tolerance	Success Ratio	Best Solution	Worst Solution
SPSO	Synthetic (without Noise)	10%	0.60	1.4377×10^{-5} 36.76 (rmse=0.0038)	1.12×10^{-5} 42.84 (rmse=0.01255)
	Synthetic (5% Noise)	15%	0.30	1.33×10^{-5} 37.2999 (rmse = 0.0145)	1.223×10^{-5} 141.999 (rmse = 0.0293)
HBBBC	Synthetic (without Noise)	10%	0.80	1.525×10^{-5} 34.0 (rmse=2.31×10^{-8})	1.2293×10^{-5} 42.89.23 (rmse=0.00891)
	Synthetic (5% Noise)	15%	0.50	1.363×10^{-5} 37.39679 (rmse = 0.0143)	8.38×10^{-5} 63.0482 (rmse = 0.01253)
PPSO	Synthetic (without Noise)	10%	0.80	1.528×10^{-5} 34.0 (rmse=1.28×10^{-8})	1.2293×10^{-5} 42.89.23 (rmse=7.776×10^{-3})
	Synthetic (5% Noise)	15%	0.70	1.4919×10^{-5} 37.39679 (rmse = 0.001366)	1.1678×10^{-5} 63.0482 (rmse = 0.00275)

noise. The t-score value is 2.31 and the result showed that the mean best point found by PPSO is significantly better.

4.3 Application to the Experimental Data

Laboratory data of [7] for diffusion and sorption of chloroform in clayey soil is used to estimate the design parameters from PPSO model. The details of soil properties and through-diffusion cell parameters used in the model are given elsewhere [4]. The best solution obtained is $D_e = 11.509 \times 10^{-5}\ cm^2/sec$ and $\alpha = 12.3507$. The theoretical profile representing the temporal variation of solute concentration in the source and collector reservoir is obtained for the model parameters estimated by the PPSO model. This is plotted in Fig. 1 along with the experimental data and the published data based on visual calibration (dotted lines) [4].

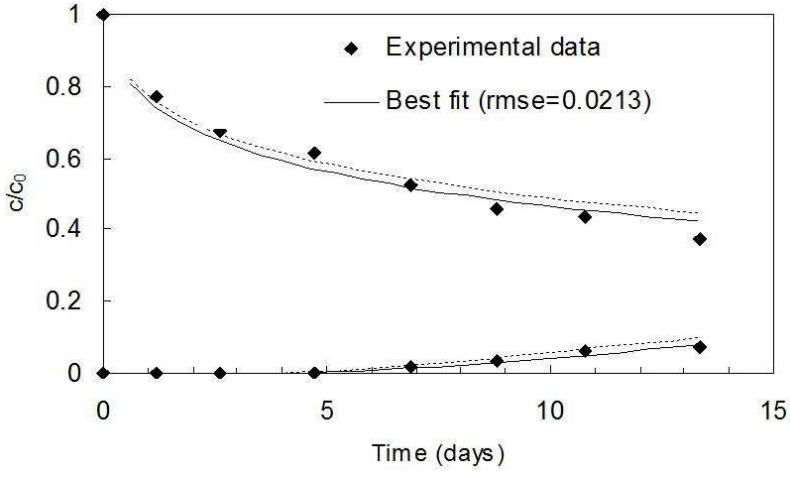


Fig. 1. The theoretical concentration data for the optimized design parameters obtained from the proposed solver for chloroform experimental data

5 Concluding Remarks

In this paper inverse model based on a new variant of pso algorithm is introduced for the parameter estimation of contaminant transport through contaminant barrier system. The location of the center of mass of the population is used in the PSO algorithm. The developed solver thus tested on synthetic data and compared with the solvers based on SPSO and HBBBC. The proposed model outperforms it's counterparts on synthetic data. Further, the model was successfully used to estimate the design parameters with good accuracy from the experimental data. Further work on improving the inverse model to enhance the success rate is in progress.

References

1. Bell, L.S.J., Binning, P.J., Kuczera, G., Kau, P.M.H.: Rigorous uncertainty assessment in contaminant transport inverse modelling: a case study of fluoride diffusion through clay liners. Journal of Contaminant Hydrology 57, 1–20 (2002)
2. Bharat, T.V.: A novel particle swarm optimizer with individual-level decision making abilities. Advances in Engineering Software (submitted)
3. Bharat, T.V., Sivapullaiah, P.V., Allam, M.M.: Swarm intelligence based inverse model for characterization of groundwater contaminant source. Electronic Journal of Geotechnical Engineering 14(B) (2009)
4. Bharat, T.V., Sivapullaiah, P.V., Allam, M.M.: Swarm intelligence-based solver for parameter estimation of laboratory through-diffusion transport of contaminants. Computers and Geotechnics 36, 984–992 (2009)
5. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evolutionary Computation 6, 58–73 (2002)
6. Erol, O.K., Eksin, I.: A new optimization method: Big bang-big crunch. Advances in Engineering Software 37, 106–111 (2006)
7. Barone, F.S., Rowe, R.K., Quigley, R.M.: A laboratory estimation of diffusion and adsorption coefficients for several volatile organics in a natural clayey soil. Journal of Contaminant Hydrology 10, 225–250 (1992)
8. Kaveh, A., Talatahari, S.: Optimal design of schwedler and ribbed domes via hybrid big bang - big crunch algorithm. Journal of Constructional Steel Research 66(3), 412–419 (2010)
9. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings, IEEE International conference on neural networks, Perth, Australia, pp. 1942–1948 (1995)
10. Peram, T., Veeramachaneni, K., Mohan, C.K.: Fitness-distance-ratio based particle swarm optimization. In: Proc. Swarm Intelligence Symp., pp. 174–181 (2003)
11. Rowe, R.K., Booker, J.R.: Pollute v.6.3.6 - 1-D pollutant migration through a nonhomogeneous soil.1983, 1990, 1994, 1997, 1998. Distributed by GAEA Environmental Engineering Ltd. (1998)
12. Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. In: Proceedings of IEEE International conference on evolutionary computation, Piscataway, NJ, USA, pp. 69–73 (1998)

Mobile Stigmergic Markers for Navigation in a Heterogeneous Robotic Swarm

Frederick Ducatelle, Gianni A. Di Caro,
Alexander Förster, and Luca Gambardella

Dalle Molle Institute for Artificial Intelligence Studies (IDSIA), Lugano, Switzerland
{frederick,gianni,alexander,luca}@idsia.ch

Abstract. We study self-organized navigation in a heterogeneous robotic swarm consisting of two types of robots: small wheeled robots, called foot-bots, and flying robots that can attach to the ceiling, called eye-bots. The task of foot-bots is to navigate back and forth between a source and a target location. The eye-bots are placed in a chain on the ceiling, connecting source and target using infrared communication. Their task is to guide foot-bots, by giving local directional instructions. The problem we address is how the positions of eye-bots and the directional instructions they give can be adapted, so that they indicate a path that is efficient for foot-bot navigation, also in the presence of obstacles. We propose an approach of mutual adaptation between foot-bots and eye-bots. Our solution is inspired by pheromone based navigation of ants, as eye-bots serve as mobile stigmergic markers for foot-bot navigation.

1 Introduction

We study how a heterogeneous robotic swarm composed of two sub-swarms can self-organize to solve a task. We are interested in mutual adaptation between sub-swarms: how can the sub-swarms adapt their behavior to each other, so that the swarm as a whole can solve the task. We focus on a navigation task, in which each sub-swarm plays a distinct role: the robots of one sub-swarm need to go back and forth between a source and a target location, while the robots of the other sub-swarm give guidance in this navigation task.

For the first sub-swarm we use wheeled robots, called foot-bots (Fig. 1(a)), and for the second flying robots that attach to the ceiling, called eye-bots (Fig. 1(b)). Both robots are under development in project Swarmanoid (http://www.swarmanoid.org). Communication between both robot types can happens through visual signalling, as they are equipped with cameras and with a multi-color LED ring around their body, while foot-bots also have a LED beacon on top. Besides this, they also exchange wireless messages locally (up to 3 m) at low bandwidth using an infrared range and bearing (IrRB) system. This system also provides relative position information.

We deploy the robots in an indoor environment. We start from a situation like the one in Fig. 1(c), where foot-bots are placed in the source location, and eye-bots are attached to the ceiling, forming a connected path between

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 456–463, 2010.

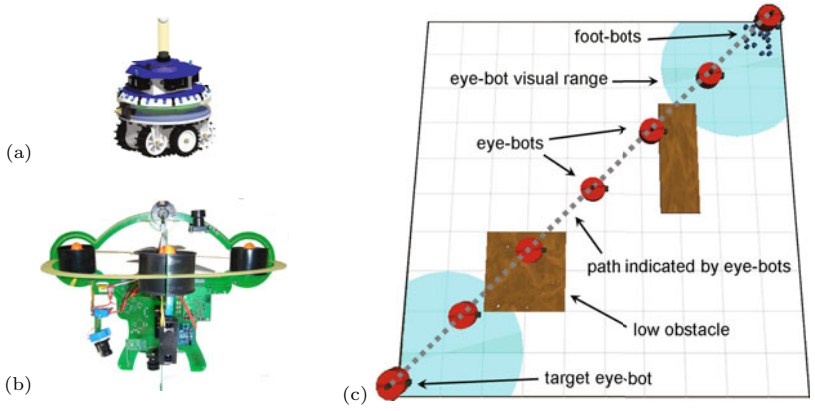


Fig. 1. (a) Example scenario, (b) Foot-bot (CAD draw), and (c) Eye-bot (prototype)

source and target (they can reach this formation using an algorithm like the one of [9]). In this initial setup, eye-bots use infrared communication among them to derive the shortest path between the two locations. They locally give directional instructions to foot-bots passing below, so that these can follow this path. The main problem is the presence of obstacles. If the environment contains obstacles (e.g., cupboards or sofas), the connected path formed by communicating eye-bots near the ceiling may pass over them. Such a path is difficult or impossible to follow for foot-bots. We investigate how eye-bots can adapt their positions and the directions they give to improve the navigability of the path they indicate.

We propose a distributed solution based on local adaptation between foot-bots and eye-bots. Foot-bots move from eye-bot to eye-bot following directional instructions received from the eye-bots they pass. Eye-bots, in turn, adapt their position and their directional instructions based on the observation of foot-bots: they move to locations where they see a lot of foot-bots, and adapt their instructions based on the directions where they see foot-bots come from. The former attracts them to areas that are navigable for foot-bots. The latter makes them indicate directions often followed by foot-bots. This way, eye-bots serve as mobile stigmergic markers for foot-bot navigation. In this sense, their role is similar to that of pheromone in ant-based navigation behavior.

2 Related Work

Swarm robotics research has mainly focused on homogeneous systems. Heterogeneous swarms have been considered only marginally, for applications like flocking [5], task allocation [6], and recruitment [8]. We know of no work where swarms of different robot types mutually adapt and jointly self-organize to solve a task.

In terms of the task, our work is related to research on self-organized path finding between a source and a target based on the pheromone guided foraging of ants in nature [4,10,11]. Various strategies have been used to implement

pheromone for the robots in the system, including light projections [10] and alcohol trails [4]. However, none of the existing work uses one swarm of robots to function as pheromone for another swarm, in the form of mobile stigmergic markers. From an application point of view, we point out the relation with work on sensor network guided robot navigation [1,7]. None of this work considers online adaptation of sensor node positions to improve navigation. Moreover, they use communication links between nodes to find navigable paths for the robot: they do not deal with the situation where obstacles block the way for the robot but not for communication, which is central for us.

3 Self-organized Path Finding

3.1 General Description

We start from a situation as shown in Fig. 1(c). In the beginning, eye-bots use network communication to derive the shortest route through the eye-bot network to the source and target. Using the relative position information given by the IrRB system, each eye-bot i derives from this routing information the directions θ_i^s towards the source and θ_i^t towards the target. These directions are broadcast locally. Foot-bots follow them, moving from eye-bot to eye-bot. When they encounter an obstacle, they perform obstacle circumnavigation. They use light signals to give information to eye-bots: to show where they are, which direction they come from, and whether they are going towards target or source.

Eye-bot actions consist in moving their position and changing their directions θ_i^s and θ_i^t (overriding the directions obtained from IrRB communication). Eye-bots move in the direction of areas where they observe foot-bots. This way, they are attracted to areas that are navigable for foot-bots and to paths that are often used by foot-bots. They also move away from nearby eye-bots, which makes them spread out and avoid collisions. Finally, they make reparatory moves when they loose network connectivity with source or target, which ensures that foot-bots can move between source and target while always staying within range of an eye-bot. Eye-bots adapt their directions θ_i^s and θ_i^t based on the direction where foot-bots going to respectively the target and the source are coming from: they assume that the direction where most foot-bots going to the target come from is a good indication of the direction to the source (and vice versa). Through their adaptations to foot-bot behavior, eye-bots serve as mobile stigmergic markers for foot-bot navigation, playing a role similar to that of pheromone in ant foraging.

3.2 Giving and Following Directional Instructions

Each eye-bot i switches on a red LED in front and a blue LED in the back, in order to show a reference direction θ_i^0. At regular intervals, i broadcasts θ_i^s and θ_i^t using the IrRB system. To get directions, a foot-bot j moves under i. It uses its camera to define θ_i^0, and reads direction θ_i^s or θ_i^t from the received wireless message. j interprets θ_i^s or θ_i^t relative to θ_i^0, and derives a travel direction θ_j^n.

After obtaining a new direction θ_j^n, j turns into that direction, and moves forward for a default distance, or until it arrives under a different eye-bot. If no other eye-bot was reached, j uses its upward camera to define the direction to the closest eye-bot, and moves there. If no eye-bot is seen, j starts a random movement. If j meets an obstacle while following θ_j^n it executes an obstacle circumnavigation behavior: it moves parallel to the obstacle, for as long as it observes the direction it wanted to go in as blocked. j uses light signals to give feedback to eye-bots. It switches on its LED beacon on top and one LED in front, to show its location and the direction it is coming from, θ_j^f. The color of the front LED indicates whether j is going to the source or the target.

3.3 Updating Eye-Bot Positions

Each eye-bot i adapts its position in three ways. The first is towards foot-bots. The second is away from other eye-bots. The third is in the direction of lost communication neighbors. The eye-bots indicating source and target never move.

When an eye-bot i observes a foot-bot j, it calculates the relative distance r_{ij} and angle α_{ij} to j in i's horizontal plane. We indicate by $\mathbf{u_{ij}} = (cos(\alpha_{ij}), sin(\alpha_{ij}))$ the unit vector in the direction of j with respect to i's frame of reference. Using $\mathbf{u_{ij}}$ and r_{ij}, eye-bot i updates a two-dimensional vector $\mathbf{p_i}$, which it uses to direct its movements. After observing j, $\mathbf{p_i}$ is updated:

$$\mathbf{p_i} = \begin{cases} \mathbf{p_i} + (1 - r_{ij})\mathbf{u_{ij}} & \text{if } r_{ij} < r^f, \\ \mathbf{p_i} + (1 - r^f)\mathbf{u_{ij}} & \text{otherwise.} \end{cases} \tag{1}$$

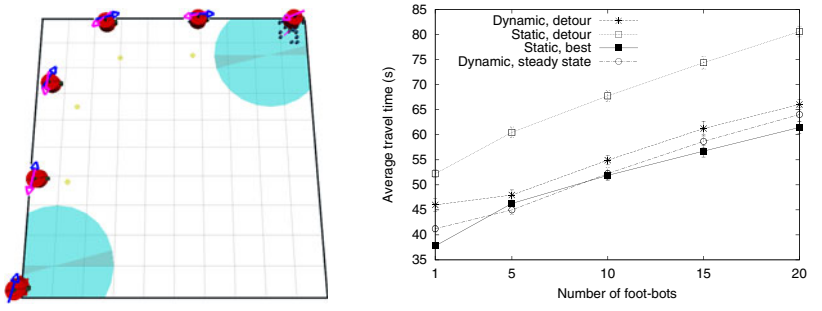
In this equation, $r^f \in [0, 1]$ is a distance threshold, which produces smaller updates for faraway foot-bots. Updating $\mathbf{p_i}$ for each foot-bot observation, eye-bot i calculates over time an aggregate of the directions in which it sees foot-bots. If foot-bots are observed more in one direction than in others, $\mathbf{p_i}$ grows in that direction. Once the magnitude of $\mathbf{p_i}$ reaches a threshold value c_p, $|\mathbf{p_i}| > c_p$, i makes a small move in the direction of $\mathbf{p_i}$. Then, $\mathbf{p_i}$ is re-initialized to $(0, 0)$.

When i observes another eye-bot k nearby, it uses the IrRB system to derive the distance r_{ik} and angle α_{ik} to k. $\mathbf{u_{ik}} = (cos(\alpha_{ik}), sin(\alpha_{ik}))$ is i's unit vector in the direction of k. In this case, the same movement vector $\mathbf{p_i}$ is updated:

$$\mathbf{p_i} + e(r_{ik})\mathbf{u_{ik}}, \tag{2}$$

where $e(r_{ik})$ is a staircase function that scales $\mathbf{u_{ik}}$ according to how far eye-bot k is: the closer k, the larger the scaling. This update makes $\mathbf{p_i}$ grow when two eye-bots get too close, so that they spread out and avoid collisions.

Finally, eye-bots also move to repair lost network connectivity (to retain a connected path between source and target). Network connectivity is established by running a routing algorithm in the eye-bot network (we use Bellman-Ford routing [2]). Using the relative position information returned by the IrRB communication system, the system derives the direction to the next hop on each path. When routing fails to indicate a next hop, connectivity is assumed to be lost. Then, eye-bots make a move towards the last known hop on the lost path.



(a) (b)

Fig. 2. Open space experiments: (a) start positions, (b) average travel time vs. number of foot-bots. Error bars show one standard deviation.

3.4 Updating Eye-Bot Directions

Initial values for θ_i^s and θ_i^t are based on the directions to next hops indicated by Bellman-Ford routing. Once an eye-bot starts observing foot-bots, it updates θ_i^s and θ_i^t based on the directions where observed foot-bots come from.

Internally, eye-bot i represents the direction θ_i^s with a two-dimensional vector $\mathbf{v_i^s}$, which points in the direction of θ_i^s and initially has size 1. At discrete time intervals, i defines the set V_i^t of foot-bots that are in view of its camera and that are going towards the target (i.e., coming from the source). For each foot-bot $j \in V_i^t$, it observes the direction it is coming from, θ_j^f, based on j's LED signals. It calculates $\mathbf{u_j^f} = (cos(d_j^f), sin(d_j^f))$, the unit vector in direction θ_j^f. Then, if $|V_i^t| > 0$, it updates $\mathbf{v_i^s}$ as in Eq. 3, and assigns the direction of $\mathbf{v_i^s}$ to θ_i^s.

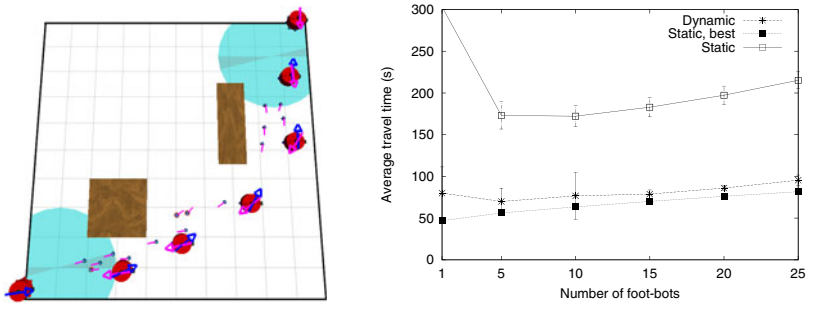
$$\mathbf{v_i^s} = a\mathbf{v_i^s} + (1-a)\sum_{j \in V_i^t} \mathbf{u_j^f}, \quad \text{where } a \in]0,1[. \tag{3}$$

4 Experimental Results

We investigate the behavior of the system through simulation tests using the Swarmanoid simulator (see `http://www.swarmanoid.org`). All experiments last 3000 seconds. We carry out 30 independent runs for each setup.

4.1 Tests in an Uncluttered Environment: Shortest Path Behavior

We first investigate uncluttered environments. We use the setup of Fig. 2(a): eye-bots start from a formation that makes a detour around the arena. Light and dark arrows above eye-bots show the directions they indicate towards respectively target and source. Results are shown in Fig. 2(b). We vary the number of foot-bots, and report the average time needed for a foot-bot to travel between source



(a) (b)

Fig. 3. Cluttered environment experiments: (a) a snapshot after 280s, (b) average travel time vs. number of foot-bots. Error bars show one standard deviation.

and target. We measured this time for the first 1000 s of simulation ("Dynamic, detour"), and between 2000 s and 3000 s, when eye-bots have had time to adapt their position ("Dynamic, steady state"). We compare to tests where eye-bots do not adapt their position or direction: tests where they remain static in the positions of Fig. 2(a) ("Static detour"), and tests where they are in a straight line between source and target ("Static best"). Results show that with our algorithm, foot-bot performance is close to that obtained over the straight path. In all cases, performance decreases with the number of foot-bots, due to congestion.

The results of Fig. 2(b) are explained by the fact that in open spaces, our algorithm lets eye-bots move to form straight paths. This is because eye-bots tend to line up with neighbors that send foot-bots to it. An eye-bot that is not lined up with its neighbors observes foot-bots more in one direction than another, and moves in that direction. E.g., for the eye-bot in the top left in Fig. 2(a), foot-bots enter its view on the right (coming from the source) or at the bottom (coming from the target). The eye-bot observes more foot-bots towards its bottom-right than towards its top-left. Its movement vector $\mathbf{p_i}$ grows towards the bottom-right, and eventually the eye-bot moves in that direction.

4.2 Experiments in a Cluttered Environment

We use the setup of Fig. 1(c). In the beginning, eye-bots indicate a straight path between source and target. Foot-bots follow this path and use obstacle circumnavigation (i.e, follow obstacle perimeters) when their way is blocked. Eye-bots move to locations where they observe foot-bots, so they tend to take place along obstacle edges. The directions indicated by eye-bots are adapted to the main movement directions of foot-bots, so they do not point towards obstacles. In the free space between obstacles, eye-bots form straight lines. Ultimately, a path of line segments connecting obstacle corners emerges. This is observed in Fig. 3(a) (the line segment above each foot-bot shows its movement direction).

We studied the effect of varying the number of foot-bots and measured the average time needed to travel between source and destination. In Fig. 3(b), we

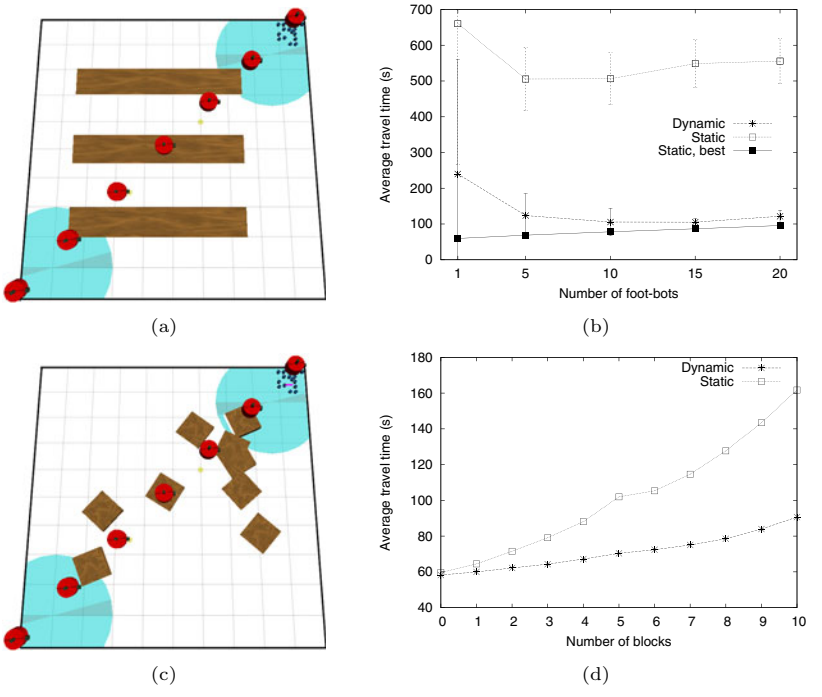


Fig. 4. Three walls experiments: (a) initial setup, (b) travel time vs. number of foot-bots. Random obstacles: (c) 10 block example, (d) travel time vs. number of blocks.

show results for tests where eye-bots use our adaptive behavior ("Dynamic"), remain static in the positions of Fig. 1(c) ("Static"), and remain static in a pre-defined path efficient for foot-bots ("Static, best"). The plots show that the dynamic approach significantly improves foot-bot navigation efficiency compared to the initial placement, and obtains results close to those of the efficient path.

4.3 Experiments in More Complex Environments

We investigate more complex environments. A first one (Fig. 4(a)), has three large obstacles, difficult to pass for foot-bots due to their size and orientation. The results in Fig. 4(b) show that the performance of static eye-bots is a lot worse than in the tests of Sect. 4.2, confirming that this scenario is more difficult. Nevertheless, the dynamic approach gets a performance close to that of a pre-defined efficient path. Next, we place obstacles randomly. These obstacles are blocks of $1 \times 1\ m^2$. We use 0 up to 10 blocks. Figure 4(c) shows a 10 block example. Results for our approach and for static eye-bots in a straight path are shown in Fig. 4(d). As the number of obstacles grows, the dynamic approach becomes more advantageous. Standard deviations are not shown, because differences between random scenarios lead to large variability (paired t-tests show that the dynamic approach is better in each data point, with p-values in the order of 10^{-6}).

5 Conclusions and Future Work

We have investigated a self-organized behavior of a heterogeneous robotic swarm involving mutual adaptations between the robots of two sub-swarms. We focused on a cooperative navigation task, and described a solution inspired by pheromone-based navigation of ants, whereby the robots of one sub-swarm served as mobile stigmergic markers for the other sub-swarm. In a number of experiments, we have investigated the performance of our system and have shown it can find efficient paths in a wide range of different scenarios. One of the main limitations of the system is that the initial directions obtained from IrRB communication put a constraint on the system's performance. If the initial path leads foot-bots to a location they cannot escape from using obstacle circumnavigation (e.g., a complex concave obstacle), the system might not find a way out. One solution is to let eye-bots explore different directions, and learn the best ones. In related work, we have investigated this for static eye-bots [3]. In future work we will consider this solution in combination with the mobility of the eye-bots.

Acknowledgments. This work was supported by the SWARMANOID project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission under grant IST-022888. The information provided is the sole responsibility of the authors and does not reflect the Commission's opinion. The Commission is not responsible for any use made of data appearing in this publication.

References

1. Batalin, M., Sukhatme, G., Hattig, M.: Mobile robot navigation using a sensor network. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (2004)
2. Bertsekas, D., Gallager, R.: Data Networks. Prentice Hall, Englewood Cliffs (1992)
3. Ducatelle, F., Di Caro, G., Gambardella, L.: Cooperative stigmergic navigation in a heterogeneous robotic swarm. In: Proceedings of SAB (2010)
4. Fujisawa, R., Dobata, S., Kubota, D., Imamura, H., Matsuno, F.: Dependency by concentration of pheromone trail for multiple robots. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) ANTS 2008. LNCS, vol. 5217, pp. 283–290. Springer, Heidelberg (2008)
5. Momen, S., Amavasai, B., Siddique, N.: Mixed species flocking for heterogeneous robotic swarms. In: Proceedings of IEEE Eurocon (2007)
6. Momen, S., Sharkey, A.: An ant-like task allocation model for a swarm of heterogeneous robots. In: Proceedings of SIAAS (2009)
7. O'Hara, K., Balch, T.: Pervasive sensor-less networks for cooperative multi-robot tasks. In: Proceedings of DARS (2004)
8. Pinciroli, C., O'Grady, R., Christensen, A., Dorigo, M.: Self-organised recruitment in a heterogeneous swarm. In: Proceedings of ICAR (2009)
9. Stirling, T., Wischmann, S., Floreano, D.: Energy-efficient indoor search by swarms of simulated flying robots without global information. In: Swarm Intelligence (2010)
10. Sugawara, K., Kazama, T., Watanabe, T.: Foraging behavior of interacting robots with virtual pheromone. In: Proceedings of IROS (2004)
11. Vaughan, R., Støy, K., Sukhatme, G., Mataric, M.: Whistling in the dark: Cooperative trail following in uncertain localization space. In: Proc. of Autonomous Agents (2000)

Motif Finding Using Ant Colony Optimization

Salim Bouamama[1], Abdellah Boukerram[2], and Amer F. Al-Badarneh[3]

[1] Department of Computer Science, University of M'sila, Algeria
bouamamas@gmail.com
[2] Department of Computer Science, University of Sétif, Algeria
boukerram@hotmail.com
[3] Jordan University of Science and Technology, Irbid, Jordan
amerb@just.edu.jo

Abstract. A challenging problem in molecular biology is the identification of the specific binding sites of transcription factors in the promoter regions of genes referred to as motifs. This paper presents an Ant Colony Optimization approach that can be used to provide the motif finding problem with promising solutions. The proposed approach incorporates a modified form of the Gibbs sampling technique as a local heuristic optimization search step. Further, it searches both in the space of starting positions as well as in the space of motif patterns so that it has more chances to discover potential motifs. The approach has been implemented and tested on some datasets including the Escherichia coli CRP protein dataset. Its performance was compared with other recent proposed algorithms for finding motifs such as MEME, MotifSampler, BioProspector, and in particular Genetic Algorithms. Experimental results show that our approach could achieve comparable or better performance in terms of motif accuracy within a reasonable computational time.

Keywords: Bioinformatics, Ant Colony Optimization, Motif Finding, Metaheuristics

1 Introduction

Finding the location of the common motif, shared by a set of DNA sequences, in each sequence has became a fundamental problem in bioinformatics with important applications in locating regulatory sites and drug target identification [14]. The motif finding problem has been formally considered as a difficult pattern recognition problem. Most developed motif finding algorithms use either approximate or heuristic techniques to obtain near optimal solutions at relatively low computational cost. Some of them carry out the search in the space of possible starting positions, whereas others search in the space of all possible motifs based on a given model. Recent researches covering most of the relevant techniques and approaches for motif finding, as well as several of the benchmarks algorithms included in this work can be found in [3,19].

Moreover, bio-inspired algorithms and other metaheuristics have been also proposed. Examples of these algorithms include genetic algorithms [12,2,8], genetic programming [15], and simulated annealing [9]. Although these methods

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 464–471, 2010.

have been shown to generate acceptable results in terms of the quality of the solutions found, the motif finding problem is still unsolved. One solution technique to respond to this challenge is a swarm-based approach, a natural metaphor, called Ant Colony Optimization (ACO) [5,4,17]. ACO is a population-based stochastic search method inspired by the foraging behavior of ant colonies. This metaheuristic has been used successfully for computing the best-known solutions for a wide range of combinatorial optimization problems (See [5] for more details). In [11], an ACO algorithm was developed to find a set of better initial positions for the Gibbs sampler (GS) [10] in order to improve its efficiently in term of time computing time and score. However, it does not incorporate any form of heuristic information. Moreover, a specific ant colony system was used for predicting the MHC class II binders [7].

In this study, we present a new motif finding approach based on ant colony optimization called MFACO that combines $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System [17], one of the best performing variants of ACO metaheuristic, with a modified form of the original GS playing the role of a local heuristic optimization step since most ACO algorithms developed in literature incorporate a particular local optimizer to improve the produced solutions. Unlike some other motif finding techniques, MFACO searches both in the space of starting positions as well as in the space of motif patterns. Due to this feature, it has more chances to find potential motif patterns. Although our approach is also valid for protein sequences, we apply it only to DNA sequences. The rest of the paper is organized as follows. In Section 2, the motif finding problem is formally introduced. Section 3 presents the main contribution of this paper. Section 4 describes the conducted experiments. Finally, concluding remarks given in Section 5.

2 The Motif Finding Problem

A formal description of this problem can be viewed as follows. Given a set of DNA sequences $S = S_1, S_2, ..., S_N$ of common length W[1]. Find the promising motif pattern $X = x_1x_2...x_i...x_l$ of length l, $x_i \in \{A, T, C, G\}$ and the starting locations of its occurrences on all sequences in S. The selection of a particular motif pattern is based on a defined score function that measure the similarity between the motif pattern and its occurrences. There are several methods for scoring a motif pattern. Our proposed approach uses *consensus score* [6] and *information content* [2,16] as score functions. To illustrate how to compute these score functions, consider a candidate motif pattern that can be generated by choosing a random position from each sequence. Then, the patterns starting at these positions are aligned to form an $N \times l$ alignment matrix.

Therefore, the candidate motif pattern can be represented by a count-based profile C where $C(i, j)$ is the count of nucleotide i on the column j of the alignment matrix and its corresponding consensus score (CSc) is defined as:

[1] Just for the purpose of simplicity the set of sequences tested in our algorithm are with the same length.

$$CSc = \sum_{j=1}^{l} (\max_{i \in \{A,T,C,G\}} (C(i,j)) \tag{1}$$

The information content (IC) score function can be easily computed as follows:

$$IC = \sum_{j=1}^{l} \sum_{i \in \{A,T,C,G\}} Q(i,j) \cdot \log_2 \frac{Q(i,j)}{B_0(i)} \tag{2}$$

Where each element $Q(i,j)$ indicates the frequency of the nucleotide i to be in position j of the motif pattern and $B_0(i)$ denotes its background frequency, i.e. the observed frequency of nucleotide i overall all sequences in the dataset.

3 The Proposed Approach (MFACO)

Our approach has the same general framework of an ACO algorithm. See [5] for more details. In the next subsections, we detail each of the MFACO components.

3.1 Initialization

Choosing the appropriate graph representation for the problem to be solved is of central importance to graph-based ACO metaheuristic. In our case, we use a weighted directed graph $G(V,E)$ with V being the set of nodes and E being the set of edges. If the motif's length is l then there are $4 \times l$ nodes arranged in a grid of four rows and l columns. Each node in position (i,j), simply denoted by $node(i,j)$, is associated to nucleotide i to be in the j^{th} position of the motif. In addition, an edge $e_i(u,v)$ always exists between two nodes $node(u,j)$ and $node(v,j+1)$ where $u,v \in \{A,T,C,G\}$ and $(1 \leq j \leq l-1)$.

In MFACO, two types of pheromone trails are modeled. First, a pheromone trail τ_i^1, $i \in \{A,T,C,G\}$, is associated with each $node(i,1)$. The value τ_i^1 encodes the desirability of nucleotide i being at the first motif's position. Second, a pheromone trail τ_{uv}^j, $u,v \in \{A,T,C,G\}$, is associated with each edge $e_j(u,v)$. The value τ_{uv}^j corresponds to the desirability of nucleotides u and v being at motif's position j and $j+1$ respectively. Initially, the pheromone values are all set to a constant value.

3.2 Solution Construction

MFACO consists of a number of iterations of solution construction. Each ant incrementally builds its solution by traversing the graph to complete a tour representing one candidate motif pattern. The probability $P_i^1(t)$ of an ant selects $node(i,1)$ as the first solution component, at iteration t, can be expressed as:

$$P_i^1(t) = \frac{\left[\tau_i^1(t)\right]^{\alpha} \left[\eta_i^0\right]^{\beta}}{\sum_{u \in \{A,T,C,G\}} \left[\tau_u^1(t)\right]^{\alpha} \left[\eta_u^0\right]^{\beta}} \tag{3}$$

where η_i^0 is the heuristic information of $node(i,1)$ representing the frequency of nucleotide i in the dataset, that is, $\eta_i^0 = B_0(i)$. Besides, an ant located at $node(u,j)$ chooses to go to $node(v,j+1)$ with a probability defined as:

$$P_{uv}^j(t) = \frac{\left[\tau_{uv}^j(t)\right]^\alpha \left[\eta_{v,j}^n\right]^\beta}{\sum_{r\in\{A,T,G,C\}} \left[\tau_{ur}^j(t)\right]^\alpha \left[\eta_{r,j}^n\right]^\beta} \tag{4}$$

where $\eta_{v,j}^n$ is the heuristic information of edge $e_j(u,v)$ which is computed based on the higher-order background model introduced in [18]. Using a background model of order n means that, in addition to pheromone information, the ant's decision to add solution component (nucleotide) i in position j of the pattern motif depends on the n previous visited nucleotides during the solution construction. Doing so, the n^{th} background model B_n is based on counting the frequency of all nucleotide subsequences of length $(n+1)$ in the dataset. Let $x_1x_2\ldots x_{j-1}x_j$ be the partial motif pattern constructed by an ant and being at $node(u,v)$, i.e. $x_j = u$. If $j < n$ then $\eta_{v,j}^n = P(x_{j+1} = v | x_{j-n+1}\ldots x_{j-1}x_j)$, otherwise $\eta_{v,j}^n = B_0(v)$. After each ant builds its solution and before the pheromone update, the generated solution is improved by applying the GS technique that plays the role of a local heuristic optimization step. It differs from the original model in the following two aspects: (i) the sequence to be excluded from the DNA sample is chosen in a round robin manner but not randomly; (ii) the new starting position with the highest score is selected instead to be randomly chosen with probability proportional to its calculated score. By these two modifications, the stochastic behavior of GS is greatly reduced and it is to the ACO that the task is shifted and preserved.

3.3 Pheromone Update

The values of pheromone trail τ_i^1, at each iteration t, are updated according to:

$$\tau_i^1(t+1) = (1-\rho)\cdot\tau_i^1(t) + \Delta\tau_i^{1,best} \tag{5}$$

where ρ $(0 < \rho < 1)$ is the pheromone evaporation rate and $\Delta\tau_i^{1,best}$ is the amount of pheromone trail deposited on $node(i,1)$ given by

$$\Delta\tau_i^{1,best} = Q(i,1)\cdot CSc^{best}/(N\cdot l) \tag{6}$$

where N is the number of sequences in the dataset and CSc^{best} is the consensus score of the best solution found either in the current iteration or so far by the algorithm which is the case of MFACO. The pheromone trail τ_{uv}^j on each edge $e_j(u,v)$ is updated according to the following updating rule:

$$\tau_{uv}^j(t+1) = (1-\rho)\cdot\tau_{uv}^j(t) + \Delta\tau_{uv}^{j,best} \tag{7}$$

where

$$\Delta\tau_{uv}^{j,best} = Q^{best}(u,j)\cdot Q^{best}(v,j+1)\cdot CSc^{best}/(N\cdot l) \tag{8}$$

Similar to $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System [17], MFACO also limits the pheromone values into an interval $[\tau_{\min}, \tau_{\max}]$ with $\tau_{\max} = CSc^{best}/((1-\rho)\cdot N\cdot l)$ and $\tau_{\min} = 0$.

4 Computational Experiments

MFACO was implemented in Matlab® version 6.5. It stops either when it reaches a predetermined maximum number of iterations or when stagnation state [4] occurs, that is, all ants keep generating the same motif pattern. Our experimental results were obtained on a PC with Pentium®IV (2.4 GHz) and 256 MB of memory. The performance of the proposed algorithm is tested on the following biological samples. Firstly, a collection of three datasets, that have been used as benchmark data in [12] with 6, 9, 18 sequences respectively. Each sequence has an equal length of 3001 nucleotides. Secondly, we have also used the sample of experimentally confirmed E. coli CRP binding sites [16]. This dataset consists of 18 sequences. Each sequence has a length of 105 nucleotides and contains at least one CRP-binding site. The conserved motif was located in 23 binding sites using the DNA Footprinting method (FP), with a motif length of 22.

The results generated by MFACO were compared with those obtained using some recent algorithms previously described for finding potential motifs. The algorithms include MotifSampler (MS) [18], BioProspector (BP) [13], MEME [1], and Genetic Algorithms (GAs). We are mainly interested in GA-based approaches, such as FMGA [12] and MDGA [2], since our approach is also a population based approach that uses stochastic choices to guide its search. FMGA searches in the space of motif patterns and uses consensus score as a similarity measure, while MDGA was implemented using information content score function and searches in the space of starting positions.

First collection of datasets. For comparison reasons we use the same way of presenting the results as in [12] and the effectiveness of such approach is measured with respect to the consensus score. The computational results for MEME, MS, and BP are obtained by sending the three datasets to each approach's website to return the three best-found motifs. The motif length is set to $l = 7$ and $l = 13$. After some preliminary experiments, we found that it is suitable to initialize MFACO using the following parameter settings: $it_{\max} = 50$, $n = 6$, $m = 8$, $\alpha = 2$, $\beta = 1$, $\rho = 0.15$ and all pheromone trail values are initially set to one. Table 1 through Table 6 list the best motif patterns discovered by the investigated approaches in each dataset. In each table, the first column gives the names of the approaches including ours. The second and third columns indicate the consensus sequence and the consensus score(CSc) of the found motif respectively. The column $s = 0$, $s = 1$, and $s = 2$ denotes the total number of sequences in the dataset that completely match the corresponding motif (i.e., exact matching without any mismatch), at most one mismatch, and at most two mismatches respectively. Table 1 shows that all approaches were able to report one completely matched motif ($CSc = 42$) of length $l = 7$. MS, MEME, and FMGA return only two completely matched motifs from the three top ranked patterns. The three best motif patterns found by BP represent the same completed matched motifs. Otherwise, MFACO achieves better results than the other approaches since it can find 18 completely matched motifs from the beginning. The use of a hash matrix by MFACO facilitates the task of finding motifs that match all sequences

especially when n is set to 6. From the rest of the tables we can see also that the motif patterns with the highest accuracy are generally found by MFACO while the others fail to discover them. We should also mention that MFACO can find different potential motif patterns having the same consensus score as well as those with multiple occurrences in a single sequence. Besides, it is clear that the serial implementation of our approach takes more time but not at the cost of an exhaustive search which has to scan the set of all 4^l patterns for a motif of length l. During our experiments, at most three runs for MFACO seems to be sufficient to predict better motifs in most cases within an acceptable computational time. The average running time was about 15 seconds, 60 seconds, 30 seconds, 75 seconds, 95 seconds, and 200 seconds for the following tests (Dataset 1, $l = 7$), (Dataset 1, $l = 13$), (Dataset 2, $l = 7$), (Dataset 2, $l = 13$), (Dataset 3, $l = 7$), and (Dataset 3, $l = 13$) respectively.

E. Coli CRP Binding Sites. This dataset was used to test the performance of MFACO compared to MDGA [2]. For doing so, we adjusted MFACO to work with the information content score rather than the consensus score, which is calculated according to Eq. (2). Each zero element of the frequency-based profile $Q(i, j)$ is set to $(0.25/23)$ and $B_0(i) = 0.25$ for each nucleotide i following the same setting done in [16]. Table 7 summarizes the starting positions of motifs discovered in the CRP dataset using FP, GS, BP, and MDGA respectively. A single sequence may contain two occurrences for the same motif. Additional column ER follows each method shows the deviation of the predicted starting positions from the exact starting positions. From the presented results shown in Table 7, it is clear that all the three approaches (GS, BP, and MDGA) failed to predict the exact starting positions identified by Footprinting. However, Table 8 illustrates that MFACO is capable of finding the exact binding sites (i.e., BS3) and new binding sites BS1, BS2, and BS4 representing the same referenced motif.

5 Conclusions

The motif finding problem is one of the challenging problems in molecular biology. The goal of this study was to investigate and adapt ACO for identifying potential motifs in gene promoter regions. The proposed approach, called MFACO, use a modified version of the Gibbs sampler playing the role of local optimizer. Experimental results have shown that MFACO is able to generate better potential motif patterns in term of prediction accuracy than other methods such as GAs, MEME, MS, and BP within a reasonable computation time. Moreover, the proposed algorithm differs from other motif finding techniques by searching both in the space of starting positions and in the space of motif patterns. This combination significantly provides more chance to explore the search space. MFACO commonly maintains its diversity until a near optimal solution will be found and can be easily adapted to work with different score functions such as consensus score and information content.

Table 1. Dataset 1, $l = 7$

Method	*Consensus*	*CSc*	$s = 1$	$s = 0$
MFACO	AAAAAAA	42	6/6	6/6
	AGGAGGA	42	6/6	6/6
	AAAAAAG	42	6/6	6/6
	TAAAAAT	42	6/6	6/6
MS	GCGGGCG	42	6/6	6/6
	CGCCGCC	42	6/6	6/6
	GGGGCGG	41	6/6	5/6
BP	GCCGCCG	42	6/6	6/6
MEME	AAAAAAA	42	6/6	6/6
	TAAAAAT	42	6/6	6/6
	AAATAAA	41	6/6	5/6
FMGA	AAAAAAA	42	6/6	6/6
	AGGAGGA	42	6/6	6/6

Table 2. Dataset 1, $l = 13$

Method	*Consensus*	*CSc*	*s*=2	*s*=1
MFACO	AAAAAAAAAAAGA	76	6/6	6/6
	AAAAAAAAAAAAG	75	6/6	5/6
	AAAAAAAAAAAGT	75	6/6	6/6
	AAAAAGAAAAAGA	75	6/6	5/6
MS	CGCCGCCGCCGCC	72	6/6	4/6
	CCGCCGCCGCCGC	71	6/6	3/6
	CGGCGGGCGGGGG	70	6/6	4/6
BP	GCCGCCGCCGCCG	72	6/6	4/6
MEME	AAAAAAAAAAAGA	76	6/6	6/6
	GAGGCTGAGGCAG	71	5/6	4/6
	AAAAAAAAAAAGA	76	6/6	6/6
FMGA	AAAAAAAAAAAAA	73	6/6	4/6

Table 3. Dataset 2, $l = 7$

Method	*Consensus*	*CSc*	$s = 1$	$s = 0$
MFACO	CCCTCCT	63	9/9	9/9
	CCCTCAG	62	9/9	8/9
	GGGTTGG	62	9/9	8/9
	GAGCAGG	62	9/9	8/9
MS	CCCGGGC	61	9/9	7/9
	CCGCGCC	60	9/9	6/9
	CGGGCGC	59	9/9	5/9
BP	GAGACGG	59	9/9	5/9
	AGTAACT	58	9/9	4/9
MEME	AAAAAAA	60	9/9	6/9
	AGGAAGA	59	9/9	5/9
	TTTTTTT	58	9/9	4/9
FMGA	CCCTCCT	63	9/9	9/9
	GGGCTGG	62	9/9	8/9

Table 4. Dataset 2, $l = 13$

Method	*Consensus*	*CSc*	*s*=2	*s*=1
MFACO	GCCGGCGGGCGCC	102	8/9	4/9
	GCGGGCGGGCGCC	101	9/9	2/9
	GCCGGCGGGCGGC	100	7/9	3/9
	GCCGGAGGGCGCC	100	8/9	2/9
MS	GGCCCCCGGGCGG	101	7/9	3/9
	CCCCGCCCCCGGC	100	8/9	2/9
	GGGGGCGCCGGGG	99	7/9	4/9
BP	GGCCCCCGGGCGG	101	7/9	3/9
	GGCCCGCGGGCGG	100	6/9	4/9
	CGGCCCGGCGCGG	97	5/9	2/9
MEME	GAAAGAGAAAGGG	99	6/9	4/9
	GGGGAGGTGGAGT	96	5/9	1/9
	TTTATTTTATTTT	95	5/9	2/9
FMGA	GCGGGGCGCGGGG	101	6/9	4/9
	GGCCGGGCGCGGG	100	6/9	5/9

Table 5. Dataset 3, $l = 7$

Method	*Consensus*	*CSc*	$s = 1$	$s = 0$
MFACO	GGGGCGG	123	18/18	15/18
	CCCAGCT	123	18/18	15/18
	CCAGCTG	123	18/18	15/18
	CTGAGGC	123	18/18	15/18
MS	GGCGGGG	123	18/18	15/18
	GCGGGGC	121	18/18	13/18
	GCGCGGG	119	18/18	11/18
BP	CGCAGGC	115	18/18	7/18
	CTGTGAT	115	18/18	7/18
	CTTGAAC	114	18/18	6/18
MEME	GTCTCTA	116	17/18	9/18
	GGCTCAA	114	18/18	6/18
	TTATCAG	105	13/18	2/18
FMGA	GGTGAGG	122	18/18	14/18
	AAAAAAA	119	18/18	11/18

Table 6. Dataset 3, $l = 13$

Method	*Consensus*	*CSc*	*s*=2	*s*=1
MFACO	GGGAGGCTGAGGC	205	16/18	6/18
	CGGGAGGCGGAGG	204	15/18	7/18
	GGAGGCTGAGGCA	202	12/18	7/18
	CGGGAGGCGGGGG	201	15/18	7/18
MS	CGGGGGGCGGAGG	196	12/18	4/18
	GGGCCGGGCGCGG	195	13/18	3/18
	GGGCCGGGCGGGG	195	11/18	4/18
BP	GAAACTCCGTCTC	186	6/18	5/18
	GCGAAACCCCGTC	186	7/18	4/18
	GCGAAACTCCGTC	185	6/18	5/18
MEME	AAAAAAAAAAAAA	194	11/18	4/18
FMGA	GGGAGGCGGAGGC	201	13/18	9/18
	AAAAAACAAAAAA	196	11/18	7/18

Table 7. Performance results achieved by GS, BP, and MDGA in the CRP dataset [2]

Seq.	*FP*	*GS*	*ER*	*BP*	*ER*	*MDGA*	*ER*
1	17, 61	59	-2	63	2	62	1
2	17, 55	53	-2	57	2	56	1
3	76	74	-2	78	2	77	1
4	63	59	-4	65	2	64	1
5	50	11	-39	52	2	51	1
6	7, 60	5	-2	9	2	8	1
7	42	40	-2	26	-16	43	1
8	39	37	-2	41	2	40	1
9	9, 80	7	-2	11	2	10	1
10	14	12	-2	16	2	15	1
11	61	59	-2	63	2	62	1
12	41	47	6	43	2	42	1
13	48	46	-2	50	2	49	1
14	71	69	-2	73	2	72	1
15	17	15	-2	19	2	18	1
16	53	49	-4	55	2	54	1
17	1, 84	25	24	68	-16	56	-28
18	78	74	-4	80	2	77	1

Table 8. Performance results achieved by MFACO in the CRP dataset

Seq.	*FP*	*BS1*	*ER*	*BS2*	*ER*	*BS3*	*ER*	*BS4*	*ER*
1	17, 61	61	0	61	0	61	0	61	0
2	17, 55	55	0	55	0	55	0	55	0
3	76	76	0	76	0	76	0	76	0
4	63	63	0	63	0	63	0	63	0
5	50	50	0	50	0	50	0	50	0
6	7, 60	7	0	7	0	7	0	7	0
7	42	24	-18	42	0	42	0	42	0
8	39	39	0	39	0	39	0	20	-19
9	9, 80	9	0	9	0	9	0	9	0
10	14	14	0	14	0	14	0	14	0
11	61	61	0	61	0	61	0	61	0
12	41	41	0	41	0	41	0	41	0
13	48	48	0	48	0	48	0	48	0
14	71	71	0	71	0	71	0	71	0
15	17	17	0	17	0	17	0	17	0
16	53	53	0	53	0	53	0	53	0
17	1, 84	84	0	84	0	84	0	84	0
18	78	78	0	76	-2	78	0	76	-2

References

1. Bailey, T.L., Elkan, C.: Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In: Proc. of the 2nd Int. Conf. on Intelligent Systems for Molecular Biology, pp. 28–36. AAAI Press, Menlo Park (1994)
2. Che, D., Song, Y., Rasheed, K.: MDGA: Motif discovery using a genetic algorithm. In: Proc. of the 2005 Conf. on Genetic and Evolutionary Computation (GECCO 2005), pp. 447–452. ACM Press, Washington (2005)
3. Das, M., Dai, H.: A survey of the DNA motif finding algorithms. BMC Bioinformatics 8(suppl.7), S21 (2007)
4. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics - Part B 26(1), 29–41 (1996)
5. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
6. Jones, N.C., Pevzner, P.A.: An Introduction to Bioinformatics Algorithms. MIT Press, Cambridge (2004)
7. Karpenko, O., Shi, J., Dai, Y.: Prediction of MHC class II binders using the ant colony search strategy. Artificial Intelligence in Medicine 35(1), 147–156 (2005)
8. Kaya, M.: MOGAMOD: Multi-objective genetic algorithm for motif discovery. Expert Systems with Applications 36, 1039–1047 (2009)
9. Keith, J.M., Adams, P., Bryant, D., Kroese, D.P., Mitchelson, K.R., Cochran, D., Lala, G.H.: A simulated annealing algorithm for finding consensus sequences. Bioinformatics 18(11), 1494–1499 (2002)
10. Lawrence, C., Altschul, S., Boguski, M., Liu, J., Neuwald, A., Wootton, J.: A Gibbs sampling strategy for multiple alignments. Science 262(5131), 208–214 (1993)
11. Liao, Y.J., Yang, C.B., Shiau, S.H.: Motif finding in biological sequences. In: Proc. of 2003 Symposium on Digital Life and Internet Technologies, Tainan, Taiwan, pp. 89–98 (2003)
12. Liu, F.F., Tsai, J.J., Chen, R., Chen, S., Shih, S.: FMGA: Finding motifs by genetic algorithm. In: IEEE 4th Symposium on Bioinformatics and Bioengineering (BIBE 2004), pp. 459–466. IEEE Press, Los Alamitos (2004)
13. Liu, X., Brutlag, D.L., Liu, J.: BioProspector: Discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. In: Pac. Symp. Biocomput., pp. 127–138 (2001)
14. Pevzner, P., Sze, S.: Combinatorial approaches to finding subtle signals in DNA sequences. In: Proc. of the 8th Int. Conf. on Intelligent Systems for Molecular Biology (ISMB 2000), pp. 269–278. AAAI Press, San Diego (2000)
15. Seehuus, R., Tveit, A., Edsberg, O.: Discovering biological motifs with genetic programming. In: Proc. of the 2005 Conf. on Genetic and Evolutionary Computation (GECCO 2005), pp. 401–408. ACM Press, Washington (2005)
16. Stormo, G.D., Hartzell, G.W.: Identifying protein-binding sites from unaligned DNA fragments. Proc. Natl. Acad. Sci. 86(4), 1183–1187 (1989)
17. Stützle, T., Hoos, H.: $\mathcal{MAX}$-$\mathcal{MIN}$ ant system. Future Generation Computer Systems 16, 889–914 (2000)
18. Thijs, G., Lescot, M., Marchal, K., Rombauts, S., Moor, B.D., Rouzé, P., Moreau, Y.: A higher order background model improves the detection of regulatory elements by Gibbs Sampling. Bioinformatics 17(12), 1113–1122 (2001)
19. Tompa, M., et al.: Assessing computational tools for the discovery of transcription factor binding sites. Nature Biotechnology 23(1), 137–144 (2005)

Multiple Ant Colony System for Substructure Discovery

Oscar Cordón[1], Arnaud Quirin[1], and Rocío Romero-Zaliz[2]

[1] European Centre for Soft Computing, Mieres (Asturias), Spain
{oscar.cordon,arnaud.quirin}@softcomputing.es
[2] Dept. of Computer Science and Artificial Intelligence,
University of Granada, Granada, Spain
rocio@decsai.ugr.es

Abstract. A system based on the adaptation of the search principle used in ant colony optimization (ACO) for multiobjective graph-based data mining (GBDM) is introduced in this paper. Our multiobjective ACO algorithm is designed to retrieve the best substructures in a graph database by jointly considering two criteria, support and complexity. The experimental comparison performed with a classical GBDM method shows the good performance of the new proposal on three datasets.

1 Introduction

Graph-based data mining (GBDM) involves an effective and efficient manipulation of relational graphs towards discovering important patterns [13]. It is now an established area which allowed the solving of a significant number of problems such as analysis of micro-array data in bioinformatics, pattern discovery in a large graph representing a social network, and analysis of transportation networks, among many others [5]. Likewise, *Pareto-based multiobjective search strategies* [2] have also gained much importance in data mining and machine learning communities. That is due to the advantage for the user of retrieving a Pareto set composed of multiple non-dominated solutions with a different trade-off in the satisfaction of some conflicting learning problem objectives [11]. Nevertheless, up to our knowledge, the idea of performing GBDM within a *multiobjective optimization* (MOO) framework, which seems to be a natural extension, has not been widely explored in the specialized literature.

MOO basics are not described in this contribution due to a lack of space, but the interested reader is referred to [2,6,3]. In short, MOO problems are characterized by several conflicting objectives which have to be simultaneously optimized [2]. The goal is to find a set of solutions, described by what is called a *decision vector*, which are superior to all the reminder and equally preferable among them, because an improvement in one objective/dimension will degrade the solution in another one. Those solutions constitute the so-called Pareto-optimal set or non-dominated solution set.

This contribution is aimed to bridge the latter gap by proposing a *multiobjective ant colony optimization* (MOACO) algorithm [9] to perform multiobjective

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 472–479, 2010.

GBDM. This novel application of MOACO is however quite natural since, as described in [8], mining a graph database can be modeled as a search in the lattice of all possible subgraphs (also called substructures). Hence, considering the use of an ACO algorithm to perform GBDM is a rather meaningful idea as these family of metaheuristic approaches are based on building solutions to combinatorial optimization problems by exploring a construction graph representing the problem space. In this way, the graph database lattice itself becomes a natural representation for the construction graph.

The method introduced in this paper is based on the classical *multiple ant colony system* (MACS)[1] [1], although any other MOACO algorithm could be considered [9]. Our multiobjective graph mining algorithm, consequently named *multiple ant colony system for substructure discovery* (MACSD), is expected to optimize two conflicting goals (viz. support and complexity) during the evaluation of the discovered subgraphs. In such a way, a Pareto set of non-dominated and meaningful substructures extracted from a graph database can be found in a single run.

The good performance of MACSD will be demonstrated when benchmarking it against the classical Subdue GBDM method [4] in an experimental study considering an artificial and two real-life datasets. Subdue [4] is one of the most extended methods in the area of GBDM [13,8] for tasks as frequent substructure discovery, graph compression and hierarchical clustering. It is based on a classical beam search driven by an heuristic, the minimum description length (MDL) principle.

The paper is organized as follows. Section 2 describes our novel methodology. Section 3 shows the performance of the MACSD algorithm on the three datasets. Finally, some conclusions are given in the Section 4.

2 Our Proposal: A Multiple Ant Colony System for Substructure Discovery

In this section, we will describe the main components of our proposal, based on the MACS algorithm. To do so, the first subsection is devoted to introduce the problem representation, i.e., the mapping of the substructure mining task to a combinatorial optimization problem representation that can be used by the artificial ants to build solutions. The second subsection will review the generic structure of our MACSD algorithm. It will also describe some specific issues related to its customization to the introduced problem representation.

2.1 Problem Representation

The design decisions taken to represent the graph mining task in such a way it can be tackled by any ACO algorithm are described below. We have followed the standard nomenclature and refer the reader to Dorigo and Stützle's book [7] for more information.

[1] Notice that, the term *Multiple* in MACS refers to the handling of multiple objectives and not to the use of multiple ant colonies.

Construction graph. As said, the ACO algorithm will take advantage of the fact that substructure mining is based on exploring a graph (the lattice representing all possible substructures). Hence, the construction graph traveled by the ants, $G_C = (C, L)$, constitutes a representation of the substructure lattice $G = (N, A)$. The set of solution components C corresponds to the set of the database graph arcs A. There is one node ij in the construction graph for each of the existing arcs between every database graph node (i, j) (at most, $|C| = n^2$). The connections L fully link the components, i.e., $|L| = |C|^2$. In this way, the construction graph includes all the possible substructures of the original lattice, i.e. all substructures have at least a support of 1. A feasible solution S generated by an ant when traveling G_C is a set of arcs (solution components) of any dimension composing a connected substructure G_S.

Constraints. The constraints enforce that a valid connected substructure included of the substructure lattice is built. Hence, they will depend on the specific kind of substructures which are to be extracted from the database (for example, they will be different in case we are extracting subgraphs or subtrees). The constraints are implicitly enforced through the solution construction process followed by the ants by properly defining the feasible neighborhood N_i^k of an ant k in node i at each construction step.

Objective functions. The multiobjective substructure discovery problem deals with the maximization of the extracted substructures complexity and support. The final aim is to uncover a non-dominated solution set composed of a variety of substructures with different trade-offs between complexity and support, which is not possible if only a single-objective algorithm such as Subdue [4] is considered.

Let $S = (N_S, A_S)$ be a feasible substructure, with $N_S \subseteq N$ being its set of nodes and $A_S \subseteq A$ being its set of arcs. We can mathematically formulate our two objectives as follows:

$$f_1(S) = Complexity(S) = \frac{|N_S| + |A_S|}{|N_{G_{max}}| + |A_{G_{max}}|} \tag{1}$$

$$f_2(S) = Support(S) = \frac{\#graphs\ in\ G\ including\ S}{card(G)} \tag{2}$$

with $card(G)$ being the cardinal of the set of graphs G composing the data base and G_{max} corresponding to a graph in G having highest sum of nodes and edges.

Pheromone trails. The pheromone trails τ_{ij} have to memorize the preference of traveling to node ij in the construction graph, i.e., of adding arc (i, j) to the substructure currently built by the ant. Hence, a pheromone trail is associated to each construction graph node ij.

Heuristic information. This information is not considered in the current algorithm version.

Solution construction. Every ant produces a single solution to the problem which corresponds to a specific extracted substructure. The final approximation set P_A built by MACSD will constitute a full solution to the problem since it will provide the user with a non-dominated set of substructures with different trade-offs between support and complexity.

To do so, each ant k starts by selecting an initial construction graph node ij (i.e., an initial database graph arc (i, j)) as its first solution component s_1^k. Instead of uniformly drawing that node, we consider it to be selected according to the following probability distribution:

$$p(ij) = \frac{\tau_{ij}}{\sum_{lm \in C} \tau_{lm}} \tag{3}$$

Therefore, the most visited arcs by the ants in the previous stages of the search are most likely to be selected as initial arcs for the exploration performed by the new ants in the current iteration.

Let $S_h^k = (s_1^k, \ldots, s_h^k)$ be the partial solution (i.e., the partial substructure) built by ant k after h construction steps. Its feasible neighborhood $N(S_h^k)$ is composed of every arc (i, j) (every construction graph node ij) such that:

1. $(i, j) \notin S_h^k$.
2. Either i, j, or both of them are included in S_h^k, i.e., they are the head, the tail, or the head and the tail of some arc included in S_h^k.

2.2 Customization of Multiple Ant Colony System for Substructure Discovery

The MACS algorithm was first proposed for vehicle routing problems [1] as an extension of the classical ant colony system (ACS) [7]. To design our MACSD proposal, we have considered the original definition of MACS and have taken some additional design decisions, which are described as follows.

External Pareto archive initialization and update. We consider an initial set of random substructures of size up to $Size_{\mathrm{M}}$ nodes to constitute the initial Pareto archive. The archive is updated *after each single ant move* and the dominated solutions are removed during each update.

Modified solution construction process. We must deal with the problem of not knowing the size of the optimal solutions in advance. To do so, in each iteration, a fixed percentage γ of the ants in the colony will build their solutions from scratch, and the remaining $1 - \gamma$ ants will randomly select one solution from the current Pareto archive and will start their construction process from it. In addition, at each step, we also decide when to stop the construction process of each ant according to a probability distribution: $p_{stopping}(S^k) = step^k / Size_{\mathrm{M}}$, with $step^k$ being the number of construction steps taken by ant k in the current iteration.

Transition rule. MACSD uses a single pheromone trail matrix, τ. The following expression is considered for the transition rule:

$$ij = \begin{cases} \arg\ \max_{lm \in N(S^k)}\ \tau_{lm}, & \text{if } q \leq q_0, \\ \hat{ij}, & \text{otherwise.} \end{cases} \tag{4}$$

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}}{\sum_{lm \in N(S^k)} \tau_{lm}}, \text{ if } lm \in N(S^k), \\ 0, \qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{5}$$

Pheromone trails update. Every time an ant travels to the node ij, it performs the local pheromone update as follows: $\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0$, where ρ is the rate of pheromone evaporation.

In the original MACS algorithm, the initial value for the pheromone trails τ_0 is calculated from a set of heuristic solutions by taking their average costs, $\hat{f}^0$ and $\hat{f}^1$, in each of the two objective functions, f_1 and f_2, and applying the following expression: $\tau_0 = 1/(\hat{f}^0 \cdot \hat{f}^1)$. In our case, we have considered the use of the set of non-dominated solutions composing the initial Pareto archive P_A. τ_0 is then computed from the average values of the latter solutions in the two optimization criteria, complexity and support, $\hat{f}_1$ and $\hat{f}_2$, respectively, by using the previous MACS expression. Of course, the τ_0 value is recomputed after each Pareto archive update.

3 Experiments and Analysis of Results

In this section we analyze the behaviour of the MACSD algorithm by means of various metrics proposed in the EMO literature [3]. Firstly, we describe the datasets and the parameter values, then we report a comparison with Subdue.

3.1 Datasets

We have used three different application domains, an artificial dataset (*shapes*) and two real-world datasets: visual science maps (*scientograms*) and web pages (*www*), which are described as follows:

shapes. This dataset [4] consists of 100 randomly generated stacks of geometrical objects and has a complexity of 500 nodes, 400 directed edges, and 6 unique labels.

scientograms. This dataset [12] is comprised by 10 scientograms of the scientific production of the USA for period 1996-2005 and has a complexity of 2762 nodes, 2769 undirected edges, and 293 unique labels.

www. This real web pages database [10] is available online on the Subdue website[2] and has a complexity of 832 nodes, 885 directed edges, and 511 unique labels which include self-connection edges.

[2] http://ailab.wsu.edu/Subdue/datasets/webdata.tar.gz

3.2 Experiments

Subdue was run 3 times, each time using one of its three different criteria as a goal (namely, complexity, support, and MDL). The results of these three runs were joined in a single Pareto set approximation (only non-dominated solutions are kept). We used the default parameters but the number of solutions to be found was set to 33 for each run, in order to have a maximum of 100 generated solutions. The MACSD algorithm was run 10 times, as a consequence of being non-deterministic. Its parameter values are as follows: 3600s. of execution time, 10 ants, $Size_{\mathrm{M}} = 3$, $\tau_0 = 0.4$, $\rho = 0.2$, $q_0 = 0.2$, and $\gamma = 0.8$. For the *shapes* dataset, we set up some specific parameters: 300s. of execution time, $Size_{\mathrm{M}} = 5$, and $q_0 = 0.5$. Those parameter values were selected from a preliminary experimentation.

The comparison between the two algorithms will be developed by considering three classical evolutionary MOO performance indicators (metrics) [6,3]: the cardinality of the Pareto set approximation, the area (S) of the Pareto front approximation, and the coverage (C) of the Pareto fronts obtained by each algorithm over those obtained by the other.

3.3 Results

The results obtained in the application of our MACSD and the Subdue algorithm to the three previously described datasets are analyzed as follows:

shapes. This dataset is small enough to be checked exhaustively: 31 non-dominated substructures have to be found, corresponding to only 8 different decision vectors. Subdue finds 16 of them (with 8 different decision vectors), getting a S-metric value of 0.108. MACSD finds 21 of them for all its runs, also obtaining the 8 possible decision vectors and the same S-metric value as Subdue. The comparison between MACSD and Subdue showed that the fronts are equal (the C-metric value is 0 in every case) but MACSD achieved a better diversity of solutions.

scientograms. This real-life dataset is more complex than the *shapes* domain. The S values for MACSD (0.206 in average on the 10 runs) are better than those obtained by Subdue (0.177). The C values obtained when comparing MACSD vs. Subdue (0.94 in all cases) are significantly greater than those obtained when comparing Subdue vs. MACSD (0.361 in average), meaning that MACSD dominates more solutions from Subdue than in the opposite comparison. Subdue achieves a higher value of cardinality (35) than MACSD (10.8 in average) as a consequence of its worst convergence to the optimal Pareto front.

Nevertheless, there are two solutions found by Subdue that MACSD did not reach, corresponding to subgraphs with the smallest support and the largest complexity values. The reason probably comes from the small number of ants allocated to MACSD.

www. This real-life dataset is as complex as *scientograms* and it also contains loops over the same node. The S values for MACSD (0.00331 in all cases) are better than that obtained by Subdue (0.00258). The C values achieved when comparing MACSD vs. Subdue (0.875 in all cases) are much greater than those obtained when comparing Subdue vs. MACSD (0.4 in all cases), meaning that MACSD dominates more solutions from Subdue than in the opposite comparison. Again, Subdue gets a higher value of cardinality (8 vs. 5) due to its worst convergence to the Pareto-optimal front. There is one solution found by Subdue that MACSD cannot reach, probably for the same reason as before.

Finally, a graphical representation of the aggregated Pareto front approximations found for each dataset is shown in Fig. 1. Although we clearly identify the said three solutions (two in *scientograms* and the other in *www*) generated by Subdue which dominate their MACSD counterparts (see the left-most extent of the Pareto fronts), MACSD extracted better Pareto fronts for both domains and found bigger substructures than Subdue for the other extent where the largest possible support value substructures are located.

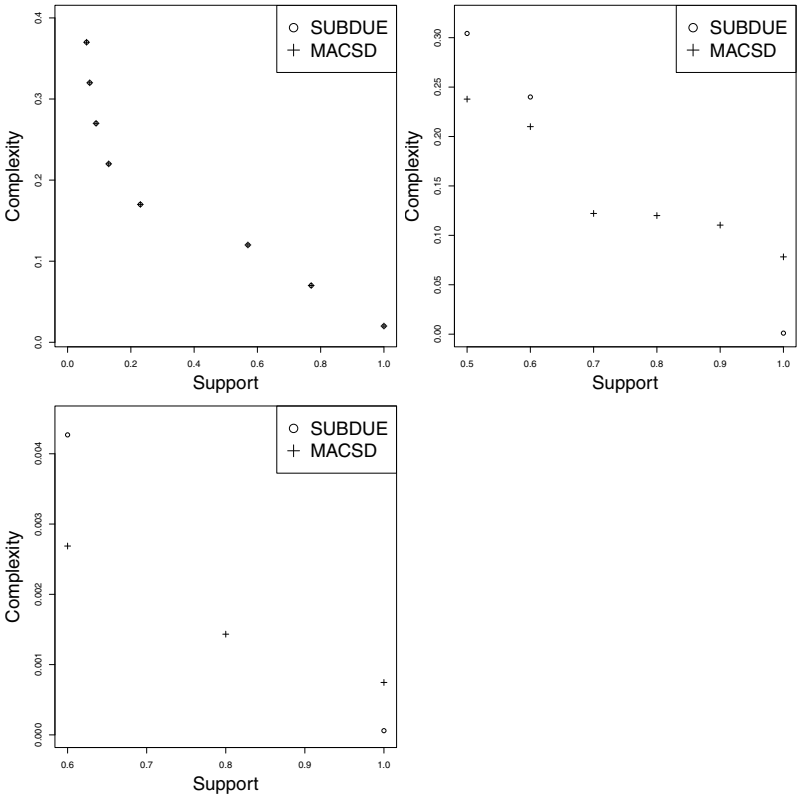


Fig. 1. Graphical representations of the Pareto front approximations for the *shapes*, *scientograms*, and *www* datasets (top left, top right, and bottom, respectively)

4 Conclusion

In this paper, we have shown how the search principle used by ACO can be naturally adapted to perform graph mining. Besides, it has been demonstrated that its combination with a MOO design (e.g. MACS) in a MOACO-based GBDM algorithm designed to retrieve the best substructures from a graph database by jointly considering the support and the complexity can report an outstanding performance. The proposed method, called MACSD, has outperformed the classical Subdue GBDM algorithm on three different datasets. As future works, we plan to design a better heuristic information definition and to test more MOACO schemes.

Acknowledgments. This work has been supported in part by the Spanish Ministry of Science and Technology under project TIN2009-07727, including EDRF fundings.

References

1. Barán, B., Schaerer, M.: A multiobjective ant colony system for vehicle routing problem with time windows. In: IASTED Conf., Innsbruck, Austria, pp. 97–102 (2003)
2. Chankong, V., Haimes, Y.Y.: Multiobjective Decision Making Theory and Methodology. North-Holland, Amsterdam (1983)
3. Coello, C.A., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-objective Problems. Springer, Berlin (2007)
4. Cook, D., Holder, L.: Graph-based data mining. IEEE Intelligent Systems 15, 32–41 (2000)
5. Cook, D., Holder, L. (eds.): Mining graph data. Wiley, London (2007)
6. Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley, Chichester (2001)
7. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
8. Fischer, I., Meinl, T.: Graph based molecular data mining - an overview. In: IEEE Int. Conf. on Systems, Man and Cybernetics., vol. 76, pp. 4578–4582 (2004)
9. García Martínez, C., Cordón, O., Herrera, F.: A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. European Journal of Operational Research 180(1), 116–148 (2007)
10. Gonzalez, J.: Empirical and Theoretical Analysis of Relational Concept Learning Using a Graph Based Representation. Ph.D. thesis, Department of Computer Science & Engineering, University of Texas, Arlington, USA (2001)
11. Jin, Y., Sendhoff, B.: Pareto-based multi-objective machine learning: An overview and case studies. IEEE Trans. Syst., Man, Cybern. C, Appl. Rev. 38, 397–415 (2008)
12. Quirin, A., Cordón, Ó., Vargas-Quesada, B., Moya-Anegon, F.: Graph-based data mining: A new tool for the analysis and comparison of scientific domains represented as scientograms. Journal of Informertics 4(3), 291–312 (2010)
13. Washio, T., Motoda, H.: State of the art of graph-based data mining. SIGKDD Explorations 5(1), 59–68 (2003)

Opportunistic Ant-Based Path Management for Wireless Mesh Networks

Laurent Paquereau and Bjarne E. Helvik

Centre for Quantifiable Quality of Service in Communication Systems*,
Norwegian University of Science and Technology, Trondheim, Norway
{laurent.paquereau,bjarne}@q2s.ntnu.no

Abstract. This paper introduces opportunistic ant-based path management for wireless mesh networks. The idea is to take advantage of the broadcast nature of the wireless medium by deferring the selection of the next-hop to the receivers of an ant. From an Ant Colony Optimization (ACO) viewpoint, the main shift is that the probabilistic forwarding decision rule is no longer executed locally at the transmitting node and in the spatial domain, but in a distributed manner among the potential forwarders and in the time domain. Early simulation results indicate that this approach improves the performance of the system in terms of convergence time, ability to adapt to changes and overhead.

1 Introduction

Path management in dynamic telecommunication networks is one of the successful applications of the Ant Colony Optimization (ACO) meta-heuristic [1,2]. In this context, ants are control packets that are used to repeatedly sample paths between source and destination pairs. Ants usually have a two-phase life cycle. On its way to the destination, a *forward ant* incrementally builds a path applying, at each node, a probabilistic forwarding decision rule based on the local selection probabilities. The selection probabilities are derived from the local pheromone trail values and, possibly, local heuristic values. Once it has reached the destination, the ant turns around and backtracks. On the way back, the *backward ant* deposits pheromones and triggers pheromone trail evaporation at each node along the sampled path. For such a system, in addition to the number of iterations and the quality of the solution, which are typically used for ACO systems, the time needed to adapt to changes and the overhead in terms of number of packets are important performance metrics.

Ant-based path management systems were first developed for wired networks and later also targeted multi-hop wireless networks, in particular Mobile Ad-hoc NETworks (MANETs). However, MANETs do not lend themselves so well to the application of ACO principles. ACO systems are intrinsically proactive, whereas frequent topology changes caused by node mobility and limited bandwidth call

* "Centre of Excellence" appointed by The Research Council of Norway, funded by the Research Council, NTNU, UNINETT and Telenor. http://www.q2s.ntnu.no.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 480–487, 2010.

for reactive strategies. Hence, most of the ACO systems proposed for path management in MANETs are intermediate between ACO systems and traditional reactive MANET routing protocols. Most of the work so far has consisted in combining elements from both worlds. Comparatively, little attention has been given to how to adapt the ACO primitives to the specificities of wireless communication. Indeed, their execution is often kept identical to what it is in the original ant system developed for wired networks.

In this paper, ant-based path management is applied to Wireless Mesh Networks (WMNs). WMNs are multi-hop wireless networks composed of static and grid-powered nodes that form a backhaul and provide Internet connectivity through a limited number of gateways. The objective is not to bear the comparison with state-of-the-art protocols, but rather to study how the execution of ACO primitives can be adapted to the specificities of wireless communication. ACO systems such as AntNet [3] or AntHocNet [4] all apply the probabilistic forwarding decision rule at the transmitting node. The decision rule is executed in the spatial domain, meaning that the transmitter chooses one of its neighbours and sends the ant to this neigbour. The approach presented here is to instead use the selection probabilities to forward ants opportunistically. We refer to this system as Opportunistic Ant System (OAS). Note that OAS is a generic ACO system. The ideas presented here are not specific to a particular system.

Opportunistic forwarding is a recently proposed paradigm that tries to exploit the broadcast nature of the wireless medium by deferring the selection of the next-hop to the receivers [5,6]. Instead of unicasting a packet to an exclusive next-hop, the transmitter broadcasts the packet and any node that receives the packet becomes a potential forwarder. One of the key tasks of such a scheme is the selection of the actual forwarder. To our knowledge, this approach has so far only been applied to data packets. In this case, only the best forwarder should then forward the packet further towards the destination. To do so, most of the proposed solutions let the transmitter specify a list of forwarders, henceforth denoted candidate forwarders, with associated priorities. The selection of the best forwarder is achieved in time by scheduling the transmission of the packet at each candidate forwarder depending on its priority and cancelling it on overhearing the transmission by a candidate forwarder with higher priority.

The rest of this paper is organized as follows. First, OAS is presented in Section 2. Next, results of a preliminary evaluation are given in Section 3. Finally, Section 4 summarizes the paper and lists future work.

2 Opportunistic Ant System

2.1 Opportunistic Forwarding of Ants

Forward Ants. OAS applies opportunistic forwarding to the transmission of forward ants. This process is illustrated in Fig. 1. OAS operates hop-by-hop and on a per-packet basis. Each ant contains a list of candidate forwarders with associated selection probabilities set by the transmitting node before broadcasting the ant. In the example presented in Fig 1(a), the list of candidate forwarders

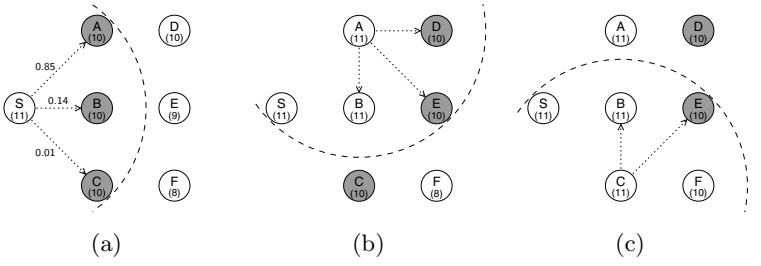


Fig. 1. Opportunistic forwarding of forward ant 10. Gray nodes represent nodes pending to forward, labeled arrows packet transmissions to candidate forwarders with their associated selection probability. The expected sequence number at each node is given in parentheses.

specified by node S contains nodes A, B and C, and their probabilities are 0.85, 0.14 and 0.01, respectively. Only nodes in this list are allowed to forward the ant further. At the reception of an ant, a candidate forwarder schedules the transmission of the ant based on the selection probability specified for this node in the ant. Unless it overhears the transmission of the ant by another candidate forwarder, it transmits the ant. In Fig. 1(b), node A forwards the ant received from node S. Node B overhears the transmission and cancels its forwarding timer. A candidate forwarder that receives an ant participates to the forwarding process, whether it forwards the ant itself or overhears the transmission.

A node participates only once in the forwarding process of a given ant. To enforce this behaviour, each ant is assigned a sequence number and each node keeps track of the next sequence number it expects. Two ants with the same sequence number are referred to as sibling ants. Only the destination node does not maintain a sequence number in order to accept sibling ants and allow for the parallel exploration of several paths. A node increments its expected sequence number on sending or overhearing the transmission of an ant. A node ignores an ant with a sequence number lower than the sequence number it expects, or if it is already pending to transmit a sibling ant and the sender is not listed as a candidate forwarder. This is the case of nodes B and E in Fig. 1(c), respectively.

As in any ACO system, the forwarding decision is probabilistic. The novelty is that the decision is executed in the time domain and in a distributed manner among candidate forwarders. Upon reception of an ant from node v, a candidate forwarder i draws a forwarding delay $d_{t_v,vi}$ from a negative exponential distribution with rate $p_{t_v,vi} \cdot \lambda$, where $p_{t_v,vi}$ denotes the selection probability of node i specified by node v. λ is the intensity of the aggregated forwarding process if all the candidate forwarders specified by v actually receives the ant. The value of λ should be chosen depending on the underlying transmission technology to assure an appropriate discrimination between the candidate forwarders. The probability that node i forwards the ant first is equal to the selection probability of node i at node v, i.e. $p_{t_v,vi}$. If all the candidate forwarders hear each others, $p_{t_v,vi}$ is then the probability that node i is the only node that forwards the ant. Hence,

the traditional ACO probabilistic forwarding decision rule is achieved among the candidate forwarders that actually received the ant. In the general case though, all the candidate forwarders may not hear each other and the probability that a given node forwards an ant depends on the actual topology and on the nodes that received the ant.

Backward Ants. Backward ants are used to update pheromone trails along the sampled paths. Therefore, a backward ant has to visit a given sequence of nodes and cannot be forwarded opportunistically. In practice however, backward ants are made similar to forward ants and are also broadcasted. Simply, the list of candidate forwarders only contains the next node to visit along the sampled path. In addition, since only one node may forward the ant, it is sent with no delay. The reason for doing so is to cancel sibling ants possibly pending at nodes around the sample path. This is particularly relevant in the neighbourhood of the destination since nodes receiving the ant at the same time as the destination would not be stopped otherwise. In the ideal case (no loss), no sibling ants should still be pending at nodes around the sampled path otherwise.

2.2 Adaptive Pruning of Forwarding Nodes

The probability that a node i is not stopped and forwards the ant may be high although the selection probability $p_{t_v,vi}$ is low. For instance, in the example shown in Fig. 1, the probability that node C forwards the ant received from node S is equal to the probability that node B does not forward the ant first, that is 0.86, although the selection probability of node C is only 0.01. This property is undesirable for two reasons: (i) it may result in a significant overhead in terms of number of sibling ants, and (ii) a low selection probability means a low intensity for the forwarding process at node i and thus longer delays. Now, the selection probability depends on the estimated goodness of using node i as a next-hop towards the destination. Hence, to remedy this problem, a "low quality next-hop" i is prevented from participating in the forwarding process by excluding it from the list of candidate forwarders at node v.

Formally, the pruning of potential forwarders at node v can be written:

$$i \in \mathcal{N}_v \Leftrightarrow p_{t_v,vi} \geqslant \varphi_{t_v,v} \tag{1}$$

where $\mathcal{N}_v \subseteq \mathcal{N}_v^*$ denotes the set of candidate forwarders, $\mathcal{N}_v^*$ the set of all potential forwarders, and $\varphi_{t_v,v}$ is the *forwarder pruning threshold*.

$$\varphi_{t_v,v} = \alpha \frac{1}{|\mathcal{N}_v^*|}(1 - E_{t_v,v}), \text{ where } E_{t_v,v} = \frac{-\sum_{\forall i \in \mathcal{N}_v^*} p_{t_v,vi} \log p_{t_v,vi}}{\log |\mathcal{N}_v^*|}. \tag{2}$$

$E_{t_v,v}$ denotes the (normalized) entropy at node v and $\alpha \in (0,1)$ is a control parameter. $E_{t_v,v}$ is a convenient scale-free measure indicating how open the search is at node v. The rationale behind this formula is to let node v self-adjust the pruning threshold relatively to the uniform probability depending on the local situation. α controls the trade-off between adaptivity/exploration and

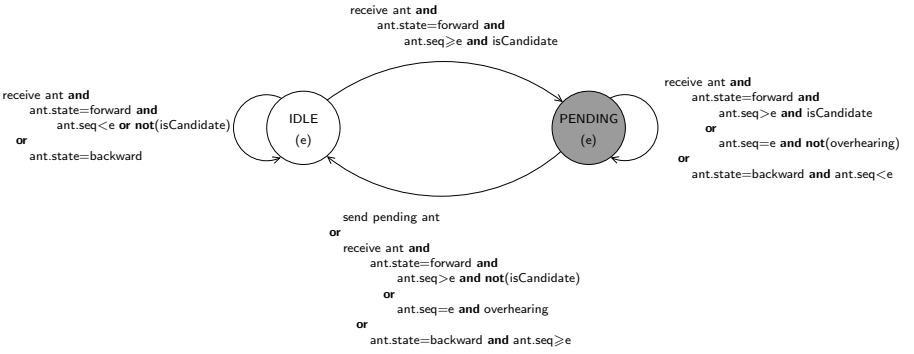


Fig. 2. Node states and internal variables

overhead. A smaller α allows the system to better adapt by allowing the system to explore alternative solutions at the cost of larger overhead. As a rule of thumb, α should be chosen at least as small as 0.1 to avoid pruning potential forwarders with a selection probability greater than an order of magnitude less than $1/|\mathcal{N}_v^*|$.

Applying adaptive pruning of forwarding nodes in the case of the example depicted in Fig. 1, $p_{SC} < \varphi_S = 0.02$ so node C would not be allowed to forward the ant received from node S.

2.3 Node States

Fig. 2 summarizes the different states of a node and the events leading to a change of state. A node is either idle or pending to forward an ant. State transitions are triggered by sending and receiving ants and depend on the sequence number of the received ant compared to sequence number expected by the node. A transition from the pending to the idle state always occurs on sending a forward ant. No transition ever occurs on sending a backward ant.

3 Preliminary Evaluation

This section presents results of a preliminary simulation study of OAS. OCEAN, an opportunistic version of the Cross-Entropy Ant System (CEAS) [7], is compared to its unicast counterpart.

3.1 Settings

The topology used for the evaluation is a 7x5 grid WMN where each node hears the transmissions from the eight nodes around. See Fig. 3 for an illustration. The simulated scenario is as follows. Nodes 2 and 18 are gateways and node 15 generates ants to establish a path to a destination node d outside the WMN through any of the gateways. The metric used is the number of hops in the

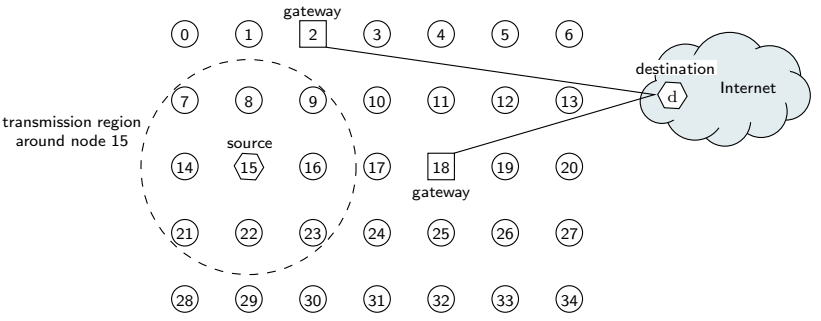


Fig. 3. 7x5 grid WMN

wireless network. From $t = 0$ s to $t = 1500$ s, the best path is via node 2. At $t = 1500$ s, node 2 is taken down and the only remaining gateway is node 18. Finally, at $t = 4000$ s, node 2 is restored.

Ants are generated at the rate of 1 per second, and 10% of the generated ants are explorer ants. Explorer ants do not use the selection probabilities at the transmitting node, but uniform probabilities. The purpose of these ants is to allow the system to discover new paths and maintain sparse pheromone trails on alternative paths providing roughly up-to-date information in case of a change. Explorers ants are also used during the initialization phase; here, from $t = 10$ s to $t = 60$ s. Finally, since explorer ants use uniform probabilities, no potential forwarders are pruned. The remaining CEAS configuration parameters are set as follows: $\beta = 0.95$ and $\rho = 0.01$. When adaptive pruning of forwarding nodes is enabled, α is set to 0.1. Ants are never retransmitted and there is no neighbourhood monitoring mechanism active. Unless otherwise specified, the results presented below are averages over 30 replications and error bars indicate 95% confidence intervals.

3.2 Results

Results presented in Fig. 4–7 demonstrate that OAS improves the performance of the system in terms of convergence time, ability to adapt to changes, and overhead. The key features of OAS explaining these results are the following.

- *Opportunistic ants may not traverse twice the same transmission region.* The size of the sample space is reduced and the feasible solutions are the ones with the lowest costs (shortest ones). Opportunistic ants, in particular explorer ants, sample shorter paths; see Fig. 6. Unicast ants sample longer paths, hence relatively poor solutions get reinforced and the system converges more slowly; see Fig. 5. In addition, the overhead, in particular of explorer ants, is significantly reduced; see Fig. 7(b).
- *The system is able to explore multiple paths simultaneously to adapt to changes by self-regulating the number of sibling ants.* In this example, when

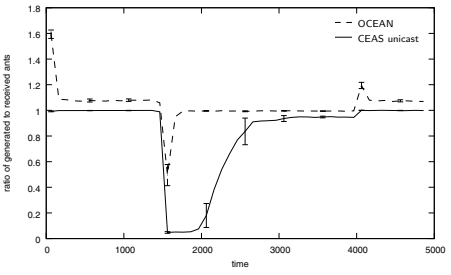


Fig. 4. Ratio of received to generated ants

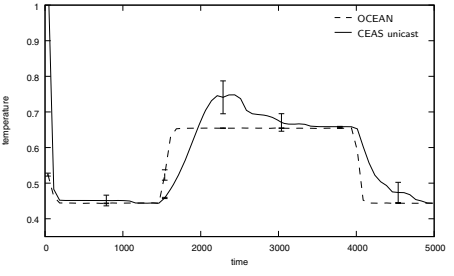
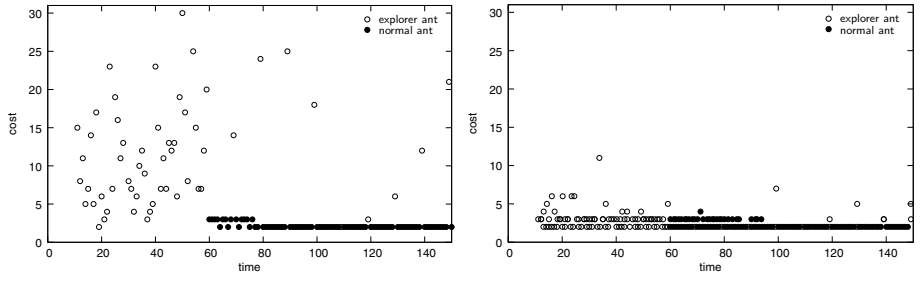


Fig. 5. Temperature at the destination[1]



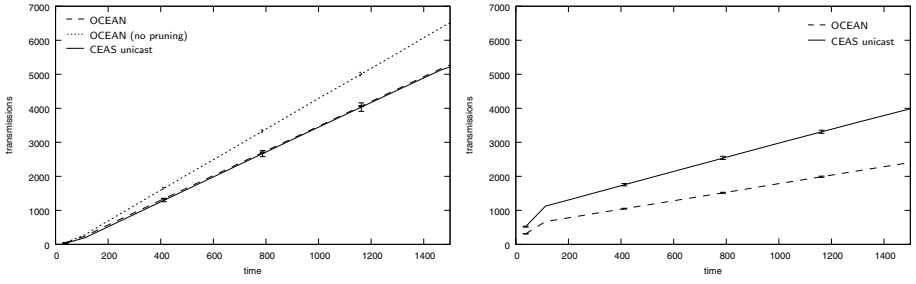
(a) CEAS unicast (b) OCEAN

Fig. 6. Costs of the sampled paths (single run)

node 2 is restored, the ratio of received to generated ants temporarily increases; see Fig. 4. More ants reach the destination and, hence, the system converges faster; see Fig. 5. The same observation can also be made for the initial convergence of the system, and this is also visible on Fig. 6(b). Explorer ants do sample two paths, but also normal ants until the system has converged. When the system has converged, adaptive pruning effectively reduces the number of sibling ants; see Fig. 7(a).

– *The system mitigates the impact of a failed transmission by allowing any candidate forwarder to send the ant further.* Immediately after node 2 is down, the ratio of received to generated ants drops because normal ants still follow the previous best path and get lost; see Fig. 4. Only explorer ants reach the destination. In the unicast case, until $t \sim 2000$ s, only explorer ants that do not travel through node 1 reach the destination. In the opportunistic case, as soon as alternative candidates are no longer pruned, normal ants eventually reach the destination allowing the system to adapt much faster.

[1] The temperature is a self-adjusting parameter of CEAS that controls the relative weights given to solutions. It depends on all the costs of the sampled paths and reflects the convergence of the system. See for instance [7] for further details.



(a) Normal ants (b) Explorer ants

Fig. 7. Cumulative number of transmissions

4 Summary and Future Work

This paper presents a novel approach to forwarding ants in WMNs taking advantage of the broadcast nature of the wireless medium and referred to as Opportunistic Ant System (OAS). OAS applies the ACO probabilistic forwarding decision rule in the time domain and in a distributed manner among the receiving nodes instead of locally at the transmitting node and in the spatial domain. A preliminary simulation study has given promising results. Compared to relying on unicast transmissions of ants, OAS exhibits better performance in terms of convergence time, adaptivity and overhead.

Future work includes testing OAS on networks of different sizes and density, studying the system when more than one source are active and comparing OAS with traditional proactive approaches.

References

1. Dorigo, M., Di Caro, G., Gambardella, L.M.: Ant Algorithms for Discrete Optimization. Artificial Life 5(2), 137–172 (1999)
2. Farooq, F., Di Caro, G.A.: Routing Protocols for Next-Generation Networks Inspired by Collective Behaviors of Insect Societies: An Overview. In: Swarm Intelligence, Natural Computing Series, pp. 101–160. Springer, Berlin (2008)
3. Di Caro, G., Dorigo, M.: AntNet: Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research 9, 317–365 (1998)
4. Ducatelle, F.: Adaptive Routing in Ad Hoc Wireless Multi-hop Networks. PhD thesis. University of Lugano, Switzerland (2007)
5. Liu, H., Zhang, B., Mouftah, H.T., Shen, X., Ma, J.: Opportunistic Routing for Wireless Ad Hoc and Sensor Networks: Present and Future Directions. IEEE Communications Magazine 48(2), 103–109 (2009)
6. Bruno, R., Nurchis, M.: Survey on diversity-based routing in wireless mesh networks: Challenges and solutions. Computer Communications 33(3), 269–282 (2010)
7. Heegaard, P.E., Wittner, O.J.: Overhead Reduction in a Distributed Path Management System. Computer Networks 54(6), 1019–1041 (2010)

Parallel Ant Colony Optimization Algorithm on a Multi-core Processor

Shigeyoshi Tsutsui[1] and Noriyuki Fujimoto[2]

[1] Management Information, Hannan University, Matsubara, Osaka, Japan
tsutsui@hannan-u.ac.jp
[2] Science, Osaka Prefecture University, Sakai, Osaka, Japan
fujimoto@mi.s.osakafu-u.ac.jp

Abstract. This paper proposes parallelization methods of ACO algorithms on a computing platform with a multi-core processor aiming at fast execution to find acceptable solutions. As an ACO algorithm, we use the cunning Ant System and test on several sizes of TSP instances. As the parallelization method, we use agent level parallelization in one colony using Java thread programming. According to the synchronization and exclusive control modes among threads, we propose three types of parallel ACO algorithms. Among them, that which we call the rough asynchronous parallel model shows the most promising results.

1 Introduction

Recently, microprocessor vendors supply processors which have multiple cores of 2, 4, or more, and PCs which use such processors are available at a reasonable cost. They are normally configured with symmetric multi processing (SMP) architecture. Since the main memory is shared among processors in SMP, parallel processing can be performed efficiently with less communication overhead among processors by using multi-thread programming.

In a previous paper, Tsutsui proposed a variant of the ACO algorithm called the cunning Ant System (cAS) [7] and evaluated it using TSP. The results showed that the cAS could be one of the most promising ACO algorithms. In this paper, we describe the parallelization of cAS on a multi-core processor and discuss the experimental results of the parallelized cAS when we apply the algorithms to solving TSP, a typical NP-hard problem in permutation domains.

Many parallel ACO algorithms have been studied [2,3,4,6]. Brief summaries can be found in [3,4]. In [6], parallel MMAS with k independent runs was studied. The most commonly used approach to parallelization is to use an island model where multiple colonies exchange information (i.e., solutions, pheromone matrix values, or parameters) synchronously or asynchronously. In [2], it is reported that the communication of the whole pheromone matrix leads to a decreased solution quality as well as worse run-time. However the approach of exchanging best-so-far solutions leads to good solution quality. In [3], a scheme in which the whole pheromone matrix is shared with two colonies using symmetric multi processing (SMP) is studied on TSP instances of hundreds to thousands. The results showed no clear advantage of parallel ACO algorithms over sequential

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 488–495, 2010.

algorithms. In [4], parallel MMAS algorithms using MPI libraries with various topologies have been intensively studied on TSP, and a clear advantage of parallel algorithms is reported.

In many of the above-mentioned studies, attention is mainly focused on the parallelization using multiple colonies. In contrast to these studies, in this study parallelization is performed at the agent (or individual) level in one colony aiming at speedup of the ACO algorithm on a computing platform with a multi-core processor.

2 A Brief Overview of *c*AS [7]

In traditional ACO based algorithms, each ant generates a candidate solution using the current pheromone densities. *c*AS exploits information from the existing solutions, which are stored in an archive. A part of each new candidate solution (cunning ant, *c-ant*) is taken from one of the existing solutions (donor ant, *d-ant*) in the archive, whereas the remainder of the new candidate solution is generated probabilistically using the pheromone densities.

Fig. 1 shows an example of generating a new solution (*c-ant*). In this example, the *c-ant* borrows part of the tour, $7 \rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow 3$, from the *d-ant* directly. The *c-ant* constructs the remainder of the tour for cities 4, 5, and 6 according to τ_{ij} probabilistically. The first position of the block (segment) of nodes is chosen randomly with uniform distribution. Length of the segment, l_s, (number of nodes) is sampled from a probability distribution to ensure that the average number of nodes present in the segment is $\gamma \times n$ where ($\gamma \in (0, 1]$) is a control parameter supplied by the user. The probability density function used for determining the value of l_s is,

$$f(l_s) = \begin{cases} \frac{1-\gamma}{n\gamma}\left(1 - \frac{l_s}{n}\right)^{\frac{1-2\gamma}{\gamma}} & \text{for } \gamma \in (0, 0.5] \\ \frac{\gamma}{n(1-\gamma)}\left(\frac{l_s}{n}\right)^{\frac{2\gamma-1}{1-\gamma}} & \text{for } \gamma \in (0, 0.5]. \end{cases} \quad (1)$$

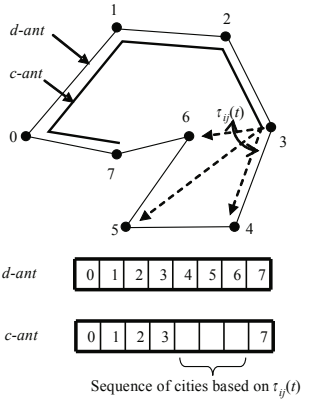


Fig. 1. *c-ant* and *d-ant*

In *c*AS, we use an elitist colony model. In this model we maintain an *archive* consisting of m candidate solutions created in the past. The kth solution in the archive at iteration t is denoted by $s_{k,t}$ ($k \in \{1, 2, \cdots, m\}$). At iteration t, a new solution is generated for each position k of the archive using the current solution in that position, $s_{k,t}$ as the donor. This candidate solution is then compared with its donor with respect to the objective function, and the better of the two is preserved at the kth position of the archive. Pheromone density $\tau_{ij}(t)$ is updated with $s_{k,t}(k \in \{1, 2, \cdots, m\})$ and $\tau_{ij}(t+1)$ is obtained as,

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t), \quad (2)$$

$$\Delta\tau_{ij}^{k}(t) = 1/C_{k,t}^{i} \text{ if } (i,j) \in s_{k,t}, \quad 0: \text{ otherwise}, \quad (3)$$

1. $t \leftarrow 0$. Initialize by setting $\tau_{ij}(t) \leftarrow C$.
2. Generate new solutions $s_{k,t}$ in the archive ($k \in \{1, 2, \cdots, m\}$)
   2.1 if $t = 0$, then
      2.1.1 $c\text{-}ant_k$ is generated according to the pheromone density $\tau_{ij}(t)$.
      2.1.2 Improve $c\text{-}ant_k$ by a local search (if we use a local search) and evaluate $c\text{-}ant_k$.
      2.1.3 Set $c\text{-}ant_k$ into archive as $s_{t,k}$.
   2.2 otherwise ($t \neq 0$)
      2.2.1 $c\text{-}ant_k$ is generated from $d\text{-}ant_k$ (i.e., $s_{k,t}$) and the pheromone density $\tau_{ij}(t)$.
      2.2.2 Improve $c\text{-}ant_k$ by a local search (if we use a local search) and evaluate $c\text{-}ant_k$.
      2.2.3 Compare $c\text{-}ant_k$ and $d\text{-}ant_k$. If $c\text{-}ant_k$ is better than $d\text{-}ant_k$, $s_{t,k}$ is replaced by $c\text{-}ant_k$, otherwise $s_{t,k}$ remains unchanged in the archive.
3. Update pheromone trail $\tau_{ij}(t)$ according to Eq. (2).
4. $t \leftarrow t + 1$.
5. If the termination criteria are met, terminate the algorithm. Otherwise, go to Step 2.

Fig. 2. Algorithm description of cAS

where the parameter $\rho(0 \leq \rho < 1)$ is the trail persistence (thus, $1 - \rho$ models the evaporation), $\Delta_{ij}^{k}(t)$ is the amount of pheromone $s_{k,t}$ puts on the edge it has used in its tour, and $C_{k,t}$ is the functional value (tour length) of $s_{k,t}$. In this updating, we keep all pheromone densities within the interval $[\tau_{min}, \tau_{max}]$ as in MMAS [5]. The algorithm of cAS is summarized in Fig. 2.

3 Parallelization of cAS on a Multi-core Processor

In this study, parallelization is performed at the agent level in one colony. Operations for each agent are performed in parallel in one colony. In our study, a set of operations for an agent is assigned to a thread in Java. Usually, the number of agents (m) may be larger than or equal to the core number of the platform, we generate n_{core} threads, where n_{core} is number of cores to be used. These threads execute operations for m agents of the cAS.

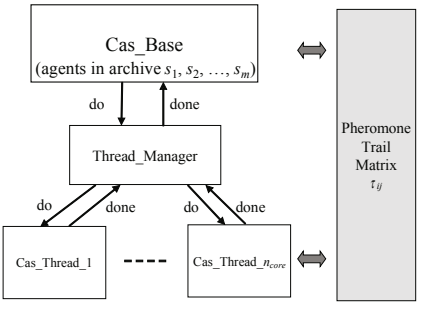


Fig. 3. Program configuration of parallel cAS

Fig. 3 shows the program configuration of the parallel cAS. The cAS_Base maintains agents in its archive and controls the entire flow of the algorithm. The Thread_Manager manages threads and assigns tasks to Cas_Thread_1, Cas_Thread_2, $\cdots$, and Cas_Thread_n_{core}. We implemented three types of parallel models as described in the following.

Model 1: The synchronous parallel cAS (SP-cAS)
As shown in Fig. 4, Step 2 of Fig. 2 can be executed independently among $s_{k,t}(k \in \{1, 2, \cdots, m\})$. Steps 3 and 4 must be excecuted after termination of all tasks in Step 2. In the synchronous parallel cAS (SP-cAS), Cas_Base requests to the Thread_Manager to assign tasks in Step 2 of Fig. 2 to Cas_Threads. Steps 3 and 4 are preformed by Cas_Base sequentially.

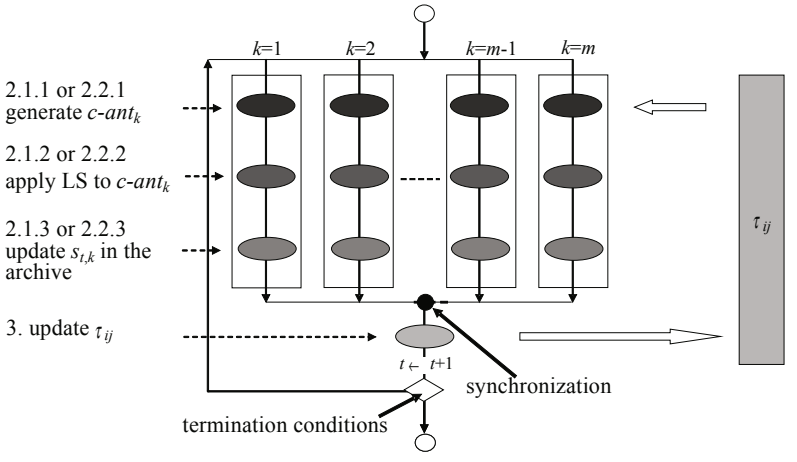


Fig. 4. Synchronous parallel cAS (SP-cAS)

Model 2: The asynchronous parallel cAS (AP-cAS)
In the synchronous parallel cAS (SP-cAS), the pheromone density updating is performed after the processing of Step 2 of Fig. 2 is completed, as in usual ACO algorithms. It may cause some waiting time in Cas_Threads when there is no agent to be processed. The asynchronous parallel cAS (AP-cAS) is intended to remove this waiting time. To attain this feature, Cas_Base requests the Thread_Manager to control Cas_Threads so that they execute all steps of Fig. 2 except for Step 1. Note that the update of the iteration counter t in Step 4 applies to each agent (each agent has its own iteration counter), and Step 5 performs only checking whether an acceptable solution has been obtained or not.

Fig. 5 shows how AP-cAS is structured. Strictly following Eq. (2), update procedure requires all archive members $s_{t,k}$ for $k = 1, 2, \cdots, m$ at the same time. But this will undermine an asynchronous execution. To perform the pheromone update asynchronously, we modified the pheromone density updating, as defined in Eq. (2) as follows.

In a Cas_Thread to which processing of agent k for $k \neq m$ is assigned, the Cas_Thread is arrowed only to increase τ_{ij}^k by Δ_{ij}^k for $(i,j) \in s_{k,t}$ in its pheromone density updating. Only a Cas_Thread to which processing of agent m is assined is allowed to complete whole updating, i.e., in addition to increasing τ_{ij}^k by Δ_{ij}^m for $(i,j) \in s_{m,t}$, multiplying the τ_{ij} by ρ, and check the max, min of τ_{ij} values. Although the above mentioned pheromone density updating is not strictly equivalent to Eq. (2), we can say that it emulates the pheromone updating process of Eq. (2) in an asynchronous processing mode.

Model 3: The rough asynchronous parallel cAS (RAP-cAS)
The rough asynchronous parallel cAS (RAP-cAS) is basically the same as AP-cAS. The difference is only in the pheromone density updating methods. In AP-cAS, the pheromone density updating is treated as a critical section. However,

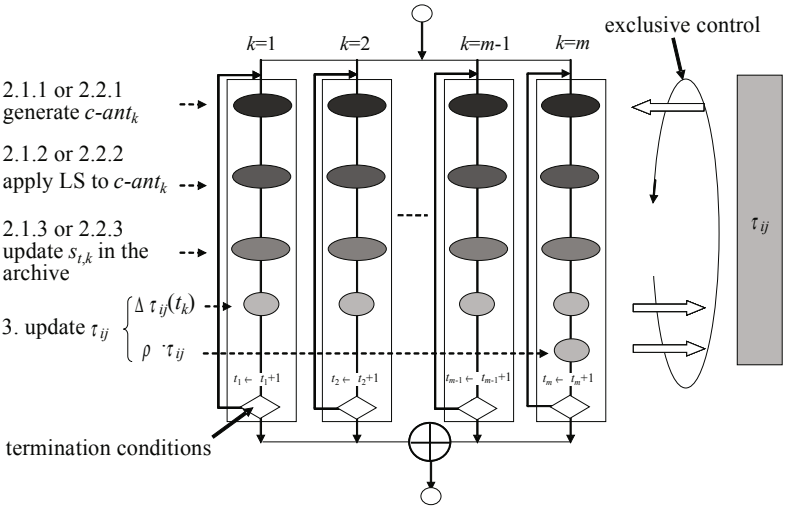


Fig. 5. Asynchronous parallel cAS (AP-cAS)

this causes some waiting time in updating τ_{ij}. In RAP-cAS, we do not treat the pheromone density updating procedure as a critical section. We allow the process run without any exclusive control accepting access conflict to τ_{ij}. Please note here, allowing access conflict to τ_{ij} never causes any fatal troubles like in a banking system.

4 Experimental Results and Analysis

4.1 Experimental Conditions and Test Instances

We used a machine which has one Intel® Core™ i7 965 (3.2 GHz) Processor, and the OS was 32-bit Windows XP. Since the processor has 4 cores, we set n_{core} to 4. The code was written in Java. We measured the performance by the number of runs in which the algorithm succeeded in finding the optimal solution ($\#OPT$) and the average time to find optimal solutions in successful runs in seconds (T_{avg}). 25 runs were performed in each experiment.

We used the following 3 classes of 9 instances; (1) small instances which are solved by combining no local search i.e., berlin52, pr76, and st70, (2) instances comprising hundreds of cities which are solved by combining 3-OPT, i.e., pcb442, att532, and rat783, and (3) instances comprising thousands of cities which are solved by combining Lin-Kernighan heuristics, i.e., pr2392, fl3795 and rl5934. Here, we used Chained LK called *Concorde TSP solver* [1]. Concorde showed good performance in our previous studies [7] on the cunning Ant System (cAS).

For Class (1) instances, population size was set to $\lceil 2 \times L/n_{core} \rceil \times n_{core}$ Maximum number of solution constructions is set to 10,000$\times L$. For Class (2)

instances, the population size is set to $\lceil (L/20/n_{core}) \rceil \times n_{core}$. Maximum execution time in one run (T_{max}) is set to 100, 200, and 300 in seconds for pcb442, att532, and rat783, respectively. For Class (3) instances, the population size is set to 4, the core count of the processor. T_{max} is set to 240, 1500, and 3000 in seconds for pr2392, fl3795 and rl5934, respectively.

Parameter γ is one of the most important parameters which affects the performance of cAS [7]. In the runs of parallel cAS, we choose γ values of 0.3, 0.2, and 0.4 for Class (1), Class (2), and Class (3), respectively.

4.2 Performance of Parallel *c*AS

Table 1 summarize the results. In all experiments, $\#OPT = 25$ was obtained though it is not written in the table. The *Speedup* indicates (T_{avg} of cAS)/(T_{avg} of parallel cAS). Please note here that this value is not the ratio of computation time in which the number of solution constructions reached a fixed value. Here we conducted a two-sided t-test of T_{avg} between RAP-cAS and SP-cAS, and between RAP-cAS and AP-cAS to show the statistical significance of the obtained results and showed the results by the p-values. In the table, we also showed the two-sided 95% confidence interval (*c-interval*) of T_{avg} of each experiment.

On Class (1) instances, we can see that the values of *Speedup* range from 2.3 to 3.1. These values are relatively smaller than the expected values of 4. Comparing parallelization methods we can see that the RAP-cAS always outperforms SP-cAS and AP-cAS. On Class (2) instances, we can see again that the RAP-cAS always outperforms SP-cAS and AP-cAS. On pcb442 and att532, super-linear speedup values (bigger than n_{core}) are observed in RAP-cAS. This is possible to occur from errors in statistics. Please recall that the *Speedup* is not the ratio of computation time of which the number of solution constructions reached a fixed value, but the ratio of time to obtain an optimal solution. As can be

Table 1. Results of parallel cAS

Class	Instances	c AS		pararllel cAS (n_{core}=4)									p-value	
				SP-c AS			AP-c AS			RAP-c AS				
		T_{avg} *2 c-interval	std *1 MSC *3	T_{avg} *2 c-interval	std *1 MSC *3	Speedup	T_{avg} *2 c-interval	std *1 MSC *3	Speedup	T_{avg} *2 c-interval	std *1 MSC *3	Speedup	SP/RAP	AP/RAP
(1) no local search	berlin52	**0.22** 0.02	0.0 27359.0	**0.10** 0.01	0.0 28313.0	**2.3**	**0.09** 0.00	0.0 27995.0	**2.5**	**0.08** 0.00	0.0 32572.0	**2.7**	0.00	0.15
	st70	**1.46** 0.13	0.0 179414.8	**0.57** 0.06	0.1 180353.2	**2.6**	**0.57** 0.11	0.3 191549.2	**2.6**	**0.47** 0.06	0.1 177596.4	**3.1**	0.04	0.12
	pr76	**1.68** 0.22	0.1 195726.8	**0.66** 0.06	0.1 201316.5	**2.5**	**0.58** 0.05	0.1 186183.9	**2.9**	**0.54** 0.05	0.1 197024.4	**3.1**	0.00	0.28
(2) 3-OPT	pcb442	**10.15** 4.67	11.3 30392.0	**2.50** 0.52	1.3 20625.8	**4.1**	**3.31** 1.10	2.7 31137.6	**3.1**	**2.01** 0.59	1.4 21940.4	**5.1**	0.21	0.05
	att532	**22.05** 6.15	14.9 33067.9	**9.11** 2.96	7.2 43506.1	**2.4**	**8.20** 2.58	6.3 42850.9	**2.7**	**5.38** 1.52	3.7 30843.8	**4.1**	0.01	0.06
	rat783	**59.87** 7.37	17.9 71630.3	**22.29** 3.80	9.2 84685.9	**2.7**	**20.54** 3.11	7.5 83457.8	**2.9**	**17.59** 2.68	6.5 82764.0	**3.4**	0.07	0.14
(3) LK	pr2392	**55.26** 11.31	27.4 43.7	**19.06** 3.59	8.7 55.8	**2.9**	**17.17** 3.51	8.5 54.3	**3.2**	**18.71** 3.90	9.4 58.9	**3.0**	0.45	0.87
	fl3795	**227.47** 56.23	136.2 42.5	**60.77** 16.54	40.1 42.7	**3.7**	**58.83** 20.47	49.6 44.6	**3.9**	**49.02** 12.36	30.0 37.0	**4.6**	0.12	0.25
	rl5934	**736.55** 143.87	348.5 205.8	**197.65** 55.08	133.4 213.4	**3.7**	**189.85** 29.73	72.0 207.0	**3.9**	**180.33** 21.34	51.7 187.2	**4.1**	0.20	0.21

*1 *std* : standard deviation

*2 *c-interval* : the two-side confidence interval of *Tavg*

*3 *MSC* : the mean number of solution constructions to find the optimum solution

understood from the *c-interval*, T_{avg} includes some errors in statistics. On Class (3) instances, RAP-*c*AS shows the best *Speedup* values except for pr2392. Again on these two instances, a super-linear speedup is observed. On pr2392, AP-*c*AS shows the best *Speedup* value, but differences among the three parallel *c*AS are minor judging from *p*-value on this instance.

Now, let us discuss two questions which arise from the results in Table 1. One question is why the *Speedup* values of Class (1) instances are relatively smaller than those of Classes (2) and (3). In Table 1, the mean number of solution constructions to find the optimum solution (MSC) is also shown. For example these values in RAP-*c*AS are 32,572.0, 21940.4, and 37.0 for berlin52 (Class (1)), pcb442 (Class (2)), and fl3795 (Class (3)), respectively. These values are equivalent to the number of activations of Cas_Threads. If we calculate $T_{thread} = (T_{avg} \times n_{core})/MSC$ T_{thread} represents the mean time of Cas_Thread's run time in one activation. Table 2 shows T_{thread} values in parallel *c*AS. For Class (1) instances, T_{thread} values are in the rage of [0.010, 0.013] milliseconds. In average, several executions of Thread_Manager are performed in one activation of Cas_Thread. As a result, on Class (1) instances, this overhead time becomes relatively large. On the other hand, on Class (2) and (3), this overhead time becomes relatively small. This fact can be the answer to the first question in the previous paragraph.

Table 2. T_{thread} values (msec)

Classes	Instances	Parallel *c* AS		
		SP-*c* AS	AP-*c* AS	RAP-*c* AS
Class (1)	berlin52	0.013	0.012	0.010
	st70	0.013	0.012	0.011
	pr76	0.013	0.012	0.011
Class (2)	pcb442	0.485	0.425	0.366
	att532	0.838	0.766	0.698
	rat783	1.053	0.985	0.850
Class (3)	pr2392	1365.5	1265.3	1269.9
	fl3795	5689.7	5275.9	5299.1
	rl5934	3704.0	3669.3	3853.2

The other question is why a super-linear speedup often occurs in RAP-*c*AS on Class (2) and (3) instances. As described before, this is possible to occur from errors in statistics. In addition to this, we can consider another possibility. In RAP-*c*AS, the pheromone update is performed without any exclusive execution control. This means the pheromone update in RAP-*c*AS causes some perturbations in τ_{ij} values. These perturbations work in a good direction in finding the solution (like mutations to the τ_{ij}). Finally, we can see that RAP-*c*AS is the most promising parallelization approach of the *c*AS, with a few exceptions. We used *c*AS for ACO algorithms. But these parallelization methods are applicable to other ACO algorithms in same manner.

5 Conclusions

In this paper we proposed parallelization methods for ACO algorithms on a computing platform with a multi-core processor aiming at fast execution to find acceptable solutions. As an ACO algorithm, we used *c*AS and tested on several sizes of TSP instances. As the parallelization method, we use agent level parallelization in one colony using Java thread programming. According to the synchronization and exclusive control modes among threads, we propose three types of parallel ACO algorithms. Among them, that which we call the rough asynchronous parallel model shows the most promising results.

As a natural progression from this study, we propose the following direction of further study: to test using machine with more cores; to use other parallel environments such as clusters, and grid; to test many-core parallel computation using GPU.

References

1. Applegate, D., et al.: ANSI C code as gzipped tar file, Concorde TSP solver (2006), http://www.tsp.gatech.edu/concorde.html
2. Benkner, S., Doerner, K., Hartl, R., Kiechle, G., Lucka, M.: Communication strategies for parallel cooperative ant colony optimization on clusters and grids. In: Dongarra, J., Madsen, K., Waśniewski, J. (eds.) PARA 2004. LNCS, vol. 3732, pp. 3–12. Springer, Heidelberg (2006)
3. Lv, Q., Xia, X., Qian, P.: A parallel aco approach based on one pheromone matrix. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 332–339. Springer, Heidelberg (2006)
4. Manfrin, M., Birattari, M., Stützle, T., Dorigo, M.: Parallel ant colony optimization for the traveling salesman problems. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 224–234. Springer, Heidelberg (2006)
5. Stützle, T., Hoos, H.: Max-min ant system. Future Generation Computer Systems 16(9), 889–914 (2000)
6. Stützle, T.: Parallelization strategies for ant colony optimization. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 722–731. Springer, Heidelberg (1998)
7. Tsutsui, S.: cAS: Ant colony optimization with cunning ants. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 162–171. Springer, Heidelberg (1996)

Particle Swarm Optimization in High Dimensional Spaces

Juan L. Fernández-Martínez[1,2,3], Tapan Mukerji[1], and Esperanza García-Gonzalo[3]

[1] Energy Resources Department, Stanford University, Palo Alto, California, USA
[2] Department of Civil and Environmental Engineering, University of California Berkeley, Berkeley, USA
[3] Department of Mathematics, University of Oviedo, Oviedo, Spain
jlfm@uniovi.es, mukerji@stanford.edu, espe@uniovi.es

Abstract. Global optimization methods including Particle Swarm Optimization are usually used to solve optimization problems when the number of parameters is small (hundreds). In the case of inverse problems the objective (or fitness) function used for sampling requires the solution of multiple forward solves. In inverse problems, both a large number of parameters, and very costly forward evaluations hamper the use of global algorithms. In this paper we address the first problem, showing that the sampling can be performed in a reduced model space. We show the application of this idea to a history matching problem of a synthetic oil reservoir. The reduction of the dimension is accomplished in this case by Principal Component analysis on a set of scenarios that are built based on prior information using stochastic simulation techniques. The use of a reduced base helps to regularize the inverse problem and to find a set of equivalent models that fit the data within a prescribed tolerance, allowing uncertainty analysis around the minimum misfit solution. This methodology can be used with other global optimization algorithms. PSO has been chosen because its shows very interesting exploration/exploitation capabilities.

Keywords: PSO, model reduction techniques, inverse problems, uncertainty.

1 Inverse Problems, Uncertainty and the Curse of Dimensionality

Inverse problems can be written in discrete form as $\mathbf{F}(\mathbf{m}) = \mathbf{d}$, where $\mathbf{m} \in \mathbf{M} \subset \mathbf{R}^n$ are the model parameters, $\mathbf{d} \in \mathbf{R}^s$ the discrete observed data, and

$$\mathbf{F}(\mathbf{m}) = (f_1(\mathbf{m}), f_2(\mathbf{m}), \ldots, f_s(\mathbf{m}))$$

is the vector field representing the forward operator and $f_j(\mathbf{m})$ is the scalar field that accounts for the j-th data. Usually $s < n$, that is, the the inverse

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 496–503, 2010.

problem has an underdetermined character. This makes the inverse problem very ill posed, that is, no unique solution exist and/or the inverse problem is very ill-conditioned. Ill-conditioning is an important issue when solving the inverse problem as an optimization problem, because noise in data is amplified back to the model parameters through the inverse forward operator, $\mathbf{F}^{-1}$.

Global optimization algorithms are a good alternative because they approach the inverse problem as a sampling problem instead of looking for the inverse operator. Also, they only need as prior information the search space of possible solutions. Typically they use as cost (or objective) function the data prediction misfit in a certain norm p : $\|\mathbf{F}(\mathbf{m}) - \mathbf{d}\|_p$. It is possible to show analytically that the topography of the cost function is that of a non-convex problem, that is, there exists a family of equivalent models that fit equally the observed data $\|\mathbf{F}(\mathbf{m}) - \mathbf{d}\|_p < tol$, that are located along flat elongated valleys. Global optimization algorithms can address the non-convexity of the cost function by sampling the family of equivalent models. Local optimization methods are not designed to approach this sampling problem, and they may even fail to find a solution without regularization. These algorithms can handle very effectively inverse problems having several thousands of parameters. The main drawback of these methods is that they are highly dependent on the initial guess and the quality of the prior information that is built in the regularization term to achieve uniqueness and stability in the inverse solution. Furthermore they do not provide any uncertainty measure around the solution of the inverse problem.

Particle swarm optimization (PSO) is a global stochastic search algorithm used for optimization motivated by the social behavior of individuals in large groups in nature [6]. Particle swarm optimization shows very impressive convergence rates when compared in real applications to other global techniques and its consistency can be related to the stability of particle trajectories through the use of stochastic stability techniques [4]. Nevertheless, one of the main limitations of global optimization algorithms (including PSO) is that they are only practical for inverse problems with small number of parameters (hundreds) and fast forward evaluations. In this paper we do not address this second issue, but we propose a simple methodology to perform sampling in high dimensional spaces through the combined use of a family of particle swarm optimizers and model reduction techniques. We show the application to reservoir parameter estimation by matching production data from a synthetic oil field. This methodology allows performing uncertainty analysis around the minimum misfit solution in order to quantify risk.

2 PSO and Model Reduction Techniques

Monte Carlo techniques and global optimization methods become unfeasible for high dimensional problems, although there are some attempts to deal with a high number of variables and fast forward evaluations. The main reason for this situation is that the base used to solve the inverse problem is the same as the one that is used to perform accurate forward predictions. We propose to use global

optimization algorithms (PSO in this case) in a reduced model space, that is, to adopt a "more-informed" base in which we solve the inverse problem. The use of model reduction techniques is based on the fact that the inverse model parameters are not independent. Conversely, there exist correlations between model parameters introduced by the physics of the forward problem $\mathbf{F}$ in order to fit the observed data. We propose to take advantage of this fact to reduce the number of parameters that are used to solve the identification problem.

To illustrate this idea let us consider an underdetermined linear inverse problem of the form $\mathbf{G}.\mathbf{m} = \mathbf{d}$ where $\mathbf{G}$ is the forward linear operator and s, n stand respectively for the dimensions of the data and model spaces. The solution to this linear inverse problem is expanded as a linear combination of a set of independent models, $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_q\}$:

$$\mathbf{m} \in \langle \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_q \rangle = \sum_{k=1}^{q} \alpha_k \mathbf{v}_k.$$

Now the inverse problem consists in finding a model in a subspace of dimension q, fulfilling:

$$GV\alpha = \mathbf{d}, \Rightarrow G\mathbf{v}_i = \mathbf{b}_i \Rightarrow B\alpha = \mathbf{d}.$$

This amounts to solving the linear system to find the weights α of the linear combination. Although this linear system might still be ill-posed, the effect of this methodology is to reduce the space of equivalent solutions. Additionally depending on the values of s and q the linear system $B\alpha = \mathbf{d}$ might be overdetermined. This methodology can be easily generalized to nonlinear inverse problems, because once the base is determined, the search is performed on the α-space. The use of a reduced set of basis vectors that are consistent with our prior knowledge allows to regularize the inverse problem and to reduce the space of possible solutions.

In this case we show an application to reservoir engineering using the PCA spatial base. Principal component analysis [7] is a well-known mathematical procedure that transforms a number of correlated variables into a smaller number of uncorrelated variables called principal components. The resulting transformation is such that the first principal component accounts for as much of the variability and each succeeding component accounts for as much of the remaining variability as possible. Usually PCA is performed in the data space, but in this case it is used to reduce the dimensionality of the model space based on a priori samples obtained from conditional geostatistical realizations that have been constrained to static data. Applied to our context, PCA consists in finding an orthogonal base of the experimental covariance matrix estimated with these prior geological models, and then selecting a subset of the most important eigenvalues and associated eigenvectors that are used as a reduced model space base. This method has been extensively used in the literature in several fields, such us weather prediction and operational oceanography, fluid dynamics, turbulence, statistics, reservoir engineering, etc. Sometimes it is also known under other terminologies such as Proper Orthogonal Decomposition or Orthogonal Empirical bases.

Let us imagine that we are able to generate an ensemble $X = [\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_q]$ of plausible scenarios that are constrained using the prior information that is at disposal. None of these scenarios obviously fit the observed data with the prescribed tolerance *tol*. Random field simulations techniques can be used for this purpose. The problem consists in finding a set of patterns $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_q\}$ that provide an accurate low dimensional representation of the original set with q being much lower than the dimension of the model space. PCA does it by diagonalizing the prior experimental covariance matrix

$$\mathbf{C}_{prior} = \frac{1}{N} \sum_{k=1}^{N} (\mathbf{m}_k - \mu) (\mathbf{m}_k - \mu)^t$$

where $\mu = \frac{1}{N} \sum_{k=1}^{N} \mathbf{m}_k$ is the experimental ensemble mean. This ensemble covariance matrix is symmetric and semi-definite positive, hence, diagonalizable with orthogonal eigenvectors $\mathbf{v}_k$, and real semi-definite positive eigenvalues. Eigenvectors $\mathbf{v}_k$ are called principal components. Eigenvalues can be ranged in decreasing order, and we can select a certain number of them to match most of the variability of the models. That is, the d first eigenvectors represent most of the variability in the model ensemble. The centered character of the experimental covariance is crucial to maintain consistency after reconstruction. Then, any model in the reduced space is represented as a unique linear combination of the eigenmodels $\mathbf{m} = \mu + \sum_{k=1}^{q} a_k \mathbf{v}_k$, where μ is the model experimental mean. The orthonormal character of the vectors provides to this base a telescopic (nested) character; that is, if we add the next eigenvector to the base, the vector will be expressed in these two bases as follows:

$$\mathbf{m} - \mu = (a_1, a_2, \ldots, a_q)_{\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_q\}} = (a_1, a_2, \ldots, a_q, 0)_{\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_q, \mathbf{v}_{q+1}\}}.$$

This property allows an easy implementation of a multi-scale inversion approach adding more eigenvectors to match higher frequencies to the model $\mathbf{m}$ as needed. To determine which level of detail we have to consider is an important question since all the finer scales might not be informed by the observed data, that is, they might belong to the null space of our local linear forward operator. By truncating the number of PCA terms that we use in the expansion we are setting these finer scales of heterogeneity (high frequencies of the model) to zero avoiding also the risk of over fitting the data. In other words, the use of a truncated PCA base provides a kind of natural smoothing of the solution.

3 Application to the History Matching Problem

Numerical models and inverse problems are very much used in reservoir characterization to improve oil production. Solving the history matching problem provides to the reservoir engineers an update of the spatial distribution of physical reservoir properties that can be used in later stages for reservoir management.

This problem has a very ill-posed character that increases with the noise level in the production data and with the reservoir complexity, that is the number of pixels needed to describe the reservoir geometry. In this case $\mathbf{F}(\mathbf{m})$ is quite complex and it is composed of multiple components: a reservoir flow simulator to predict the production data (Stanford's General Purpose Research Simulator-GPRS); a wave propagation model and inversion (diffraction tomography) to reconstruct the seismic velocities from the seismic traces measured at the boreholes; a geostatistical model to constrain the spatial structure of the reservoir, and finally a rock physics model that takes into account the facies-specific relations between porosity, permeability, saturations and elastic velocities [2]. The reservoir model is composed of 4000 cells organized in ten layers of 20x20 pixels extracted from the Stanford VI sand and shale synthetic reservoir [1].The ensemble of plausible reservoirs (one thousand) were generated by multipoint geostatistical techniques [8]. These realizations of the reservoir span what we think could be the variability of our model space. All these models are conditioned to borehole static data (facies measured at the wells).

To perform the inversion we used the cloud versions [5] of the different PSO optimizers: PSO, CP-PSO and CC-PSO [4]. In this algorithm each particle of the swarm has associated a different set of inertia and local and global acceleration constants located close to the upper border of the second order stability region of each optimizer, instead of the more common algorithm where every particle has the same set of parameters. The cloud design allows the different swarm members to find the sets of parameters that are better suited for solving each inverse problem. In the cloud some of the particles will have a more exploratory behavior while others will show a higher exploitative character. This feature helps to avoid two main drawbacks of the PSO algorithm: the tuning of the PSO parameters and the artificial clamping of the velocities. The cloud PSO

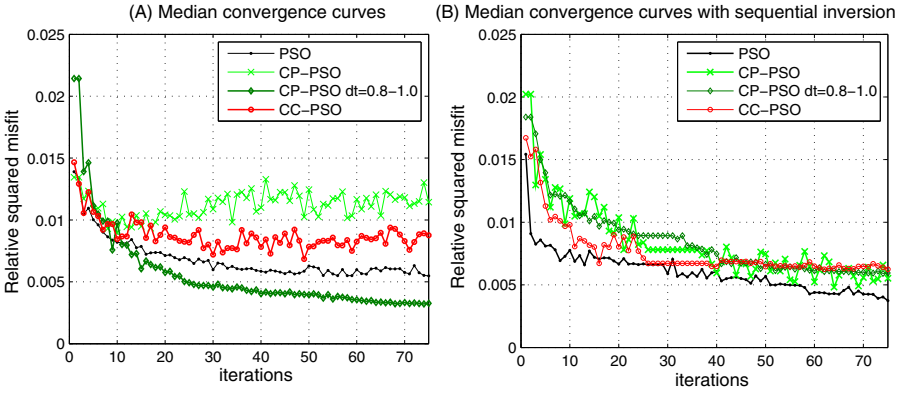


Fig. 1. A) Stanford VI reservoir (5% of Gaussian noise). Median convergence curves for different PSO versions with a swarm size of 20. B) Median convergence curves for different PSO versions using multiscale inversion with a swarm of twenty particles and ten and twenty PCA terms respectively.

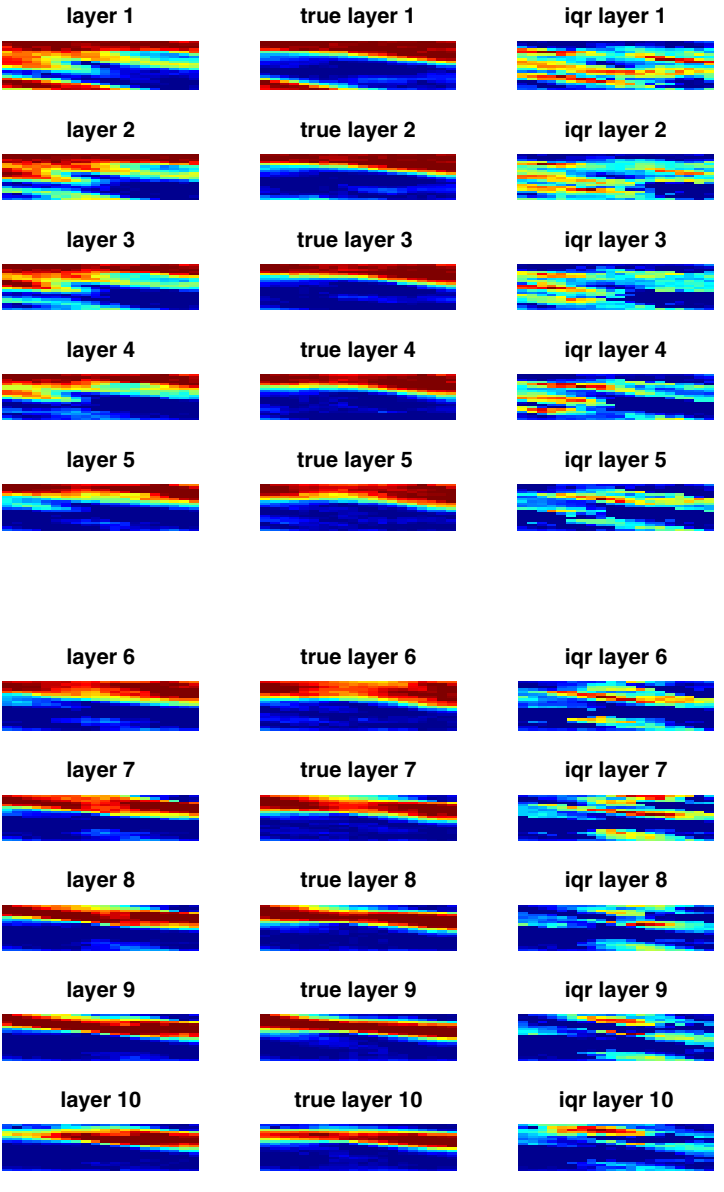


Fig. 2. Stanford VI reservoir (5% of Gaussian noise). Horizontal slices from the synthetic reservoir showing the inverted solution, true model, and uncertainty on the model parameters (inter-quartile range).

algorithms have been tested on different benchmark functions obtaining very successful results [5].

We are interested in analyzing how the PSO family members are able to optimize the history matching problem using a small number of particles. In this case we have used a swarm of 20 models over 75 iterations and the performance behavior was averaged over 10 independent simulations. The observed data is affected with 5% of Gaussian noise. In Figure 1a it can be observed that most of the family members reach the low misfit region within approximately 30 iterations. This allows us to perform posterior statistics even with a reduced number of forward runs provided that we perform enough exploration. In Figure 1a, the CP-PSO version is the member that gives the lower median curve when it is used in the lime and sand modality, that is with Δt varying between 0.8 and 1.0 with iterations [3]. With $\Delta t = 1$ the CP-PSO version is very explorative and the error does not decrease with the iterations, that is, the algorithm is exploring the region of medium misfits (0.01). The PSO and CC-PSO versions also perform well but they usually converge for this particular data set to an area of higher misfits. Figure 1b shows the median convergence curve for different family members obtained by multiscale inversion with 10 and 20 PCA terms respectively using a swarm of 20 particles. It can be observed that the convergence rate for all the PSO members gets very similar. Thus, the multiscale inversion can be performed in a natural way in the reduced model space and at each stage we are able to explore different spatial scales, adapting dynamically our model parameterization to the data resolution.

Finally our approach allows us to perform model uncertainty assessment based on the samples that have been collected on the low misfit region. Figure 2 shows for the ten layers of the Stanford VI synthetic reservoir the optimum facies model found by CP-PSO in the presence of 5% of Gaussian noise compared to the true model. We also show the uncertainty analysis deduced from the samples that can be associated to this "optimum" facies model. Although the true model is binary (sand and shale) the optimum facies model and the interquartile range show a continuous color gradation due to the truncation adopted on the PCA base. It can be observed that the inverted model approaches the true synthetic model, and although they are different, the uncertainty measures in each pixel computed in the region of square relative misfit lower than 0.010 serve to account for the difference between these models. To perform posterior statistics we also keep track of the median distance between the swarm and the global best. When this distance is smaller than a certain percentage of the initial value (5% in this case) this means that the swarm has collapsed towards the global best. Once this happens we can either stop the algorithm, or continue iterating, but in the posterior analysis we count all the particles in this collapsed swarm as one.

4 Conclusions

The combined use of high performance global algorithms such as Particle Swarm Optimization and model reduction techniques allows us to address real world applications having thousand of parameters. The use of model reduction techniques

is based on the fact that the inverse model parameters are not independent. Conversely, there exist correlations between model parameters introduced by the physics of the forward problem in order to fit the observed data. We propose to take advantage of fact to reduce the number of parameters that are used to solve the identification problem. The use of a reduced base helps to regularize the inverse problem and allows us to perform model appraisal by sampling the family of equivalent models that fit the observed data and are in accord with the prior information that is at disposal. This methodology can be used with other global optimization algorithms. PSO has been chosen because its shows very interesting exploration/exploitation capabilities.

Acknowledgments. We acknowledge the funding coming from Stanford Center for Reservoir Forecasting and Smart Fields Consortia. We also acknowledge David Echeverría (Stanford University) and Eduardo Santos (formerly at Stanford University) for lending us the forward programs to solve the history matching problem, and the Center for Computational Earth and Environmental Science at Stanford University for providing the computational resources.

References

1. Castro, S.A., Caers, J., Mukerji, T.: The Stanford VI reservoir. Tech. Rep. 18th Annual Report, Stanford Center for Reservoir Forecasting (SCRF). Stanford University, California, USA (May 2005)
2. Echeverría, D., Mukerji, T.: A robust scheme for spatio-temporal inverse modeling of oil reservoirs. In: Anderssen, R., Braddock, R., Newham, L. (eds.) 18th World IMACS / MODSIM Congress, Cairns, Australia, pp. 4206–4212 (July 2009)
3. Fernández-Martínez, J.L., García-Gonzalo, E.: The generalized PSO: a new door to PSO evolution. Journal of Artificial Evolution and Applications 2008, 1–15 (2008)
4. Fernández-Martínez, J.L., García-Gonzalo, E.: The PSO family: deduction, stochastic analysis and comparison. Swarm Intelligence 3(4), 245–273 (2009)
5. Fernández-Martínez, J.L., García-Gonzalo, E., Fernández-Muniz, Z., Mukerji, T.: How to design a powerful family of particle swarm optimizers for inverse modeling. New trends on bio-inspired computation. In: Transactions of the Institute of Measurement and Control (2010) (accepted for publication)
6. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings IEEE International Conference on Neural Networks (ICNN 1995), Perth, WA, Australia, vol. 4, pp. 1942–1948 (November-December 1995)
7. Pearson, K.: On lines and planes of closest fit to systems of points in space. Philosophical Magazine 2(6), 559–572 (1901)
8. Strebelle, S.: Conditional simulation of complex geological structures using multiple-point statistics. Mathematical Geology 34(1), 1–21 (2002)

Particle Swarm Optimization of Bollinger Bands

Matthew Butler and Dimitar Kazakov

Faculty of Computer Science, Artificial Intelligence Group, University of York, UK
{mbutler,kazakov}@cs.york.ac.uk

Abstract. The use of technical indicators to derive stock trading signals is a foundation of financial technical analysis. Many of these indicators have several parameters which creates a difficult optimization problem given the highly non-linear and non-stationary nature of a financial time-series. This study investigates a popular financial indicator, Bollinger Bands, and the fine tuning of its parameters via particle swarm optimization under 4 different fitness functions: profitability, Sharpe ratio, Sortino ratio and accuracy. The experiment results show that the parameters optimized through PSO using the profitability fitness function produced superior out-of-sample trading results which includes transaction costs when compared to the default parameters.

Keywords: particle swarm optimization, Bollinger Bands, Sharpe ratio, Sortino ratio and parameter optimization.

1 Introduction

Technical analysis is a popular technique for modeling the stock market for price level and volatility estimation. Although these techniques can be very useful they often have several parameters which have to be decided upon which greatly influence the effectiveness of the trading signals produced. One popular indicator is Bollinger Bands (BB), as proposed by John Bollinger, which are considered to be a channel indicator, where the bands create a channel around the price movements which capture the majority of the fluctuations. In work conducted by Lento et al. [2] the authors found that trading rules developed from BBs were not profitable after transaction costs were accounted for and when generating trading rules based on several indicators the models were more reliable when BBs were excluded. The authors point out that there is a lack of attention from academia in studying BBs and that there is no theoretical explanation for using the traditional parameters settings. In Leung et al. [4] the authors compared the trading rules of two channel indicators the BBs and Moving Average Envelopes (MAE), the results indicated that the MAE tended to produce more profitable trading rules in the short-term which is the preferred time-horizon for using technical trading rules. In Lento et al. [3] the authors conclude that BBs are unable to produce profitable trading rules and that they consistently underperform relative to the buy-and-hold approach (a passive investment strategy which reflects the performance of the market as a whole). These studies all have a similar approach where the size of the window to capture the information is varied and the

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 504–511, 2010.

moving average is a simple moving average, as given in equation 1. These studies also share the assumption that the bands for which the channels are constructed are equal for the upper and lower limits. Given the recent empirical evidence that BBs are not as effective as other technical indicators and that there are several parameters for which no theoretical consideration is given it seems reasonable that an advanced optimization technique such as particle swarm optimization (PSO) may be appropriate to fine tune the settings. Other work [7] revealed that there was not a global set of parameters for which BBs would perform optimally for each stock and therefore promoting an active learning approach where the parameters of the BBs can be arrived upon based on historical data and which are optimized on some relevant fitness function. Once again PSO has particular strengths for performing such an optimization approach.

2 Bollinger Bands

Bollinger Bands are a technical indicator, which creates a price channel around a moving average as depicted in figure 2. These price channels can be used to identify stocks which are overbought or oversold and therefore create trading signals for buying or selling. The three main components of a BB are:

1. An N day moving average, which creates the middle band, equation 1,

$$SMA_n(t) = \frac{\sum_{i=t-N+1}^{t} P_i}{N} \tag{1}$$

2. an upper band which, is k times above the standard deviation of the middle band, and
3. a lower band, which is k times below the standard deviation of the middle band.

The default settings for using BBs are a moving average window of 20 days and a value of k equal to 2 for both the upper and lower bands. When the price of the stock is trading above the upper band, it is considered to be overbought, and conversely, an asset which is trading under the lower band is oversold. The trading rules that can be generated from using this indicator are given by equations 2, 3:

$$Buy : P_n(t-1) < BB_n^{low}(t-1) \& P_n(t) > BB_n^{low}(t) \tag{2}$$

$$Sell : P_n(t-1) < BB_n^{up}(t-1) \& P_n(t) > BB_n^{up}(t) \tag{3}$$

Essentially, the above rules state that a buy signal is initialized when the price crosses the lower bound from below, and a sell signal when the price crosses the upper bound from above. In both cases the trade is closed out when the price crosses the middle band. As such, a trader will be taking long/short positions in the market; a long/short position is a trading technique which profits from increasing/declining asset prices.

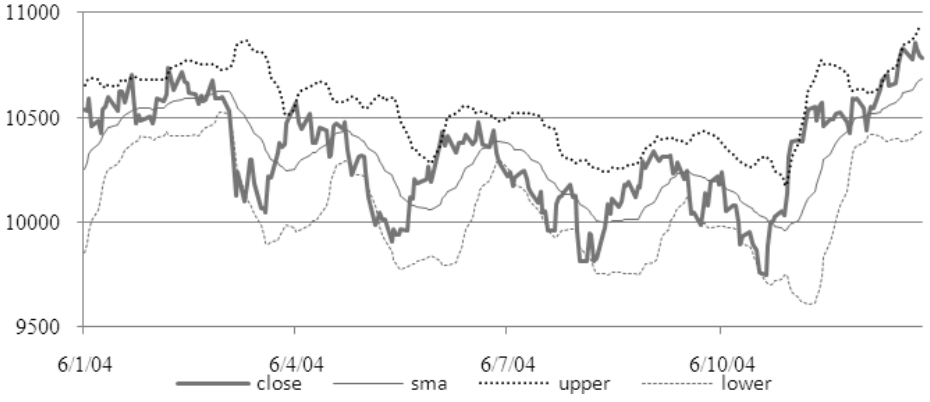


Fig. 1. A depiction of 250 days of trading for the Dow Jones Industrial Average (a market index in the USA), a 20-day moving average and the upper and lower Bollinger bands with value of k =2

3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an algorithm inspired from swarm intelligence commonly used in optimization tasks, which has had success with searching complex solution spaces, similar to the abilities of genetic algorithms (GA). PSO was chosen for this study as it has been shown to be as effective as GAs when modeling technical trading rules, as in Lee et al. [1], yet it had a much simpler implementation and arrived at a global optimum with fewer iterations. For this study the PSO was searching for the optimal values for the parameters displayed in table 1. What makes this study particular novel is that we are not making the assumption that the upper and lower bands should be equal but are allowing the PSO algorithm to search for the optimal bands independent of each other. The default settings for the type of moving average (MA) is normally a simple moving average (SMA), however it is suggested that an exponential moving average (EMA) does perform better in some markets. For this implementation we are allowing the PSO to explore this solution space where both SMA and EMA will be considered. Finally the size of the window used for calculating the moving average and the standard deviation have also been explored. The PSO algorithm was implemented with particles containing 10 dimensions with an overall swarm size of 100, the mapping of the 10-dimensional position vector to the BB parameters is provided below in figure 2.

The type of MA to use was mapped using a wrapper function which evaluated to a SMA if the particle had a value greater than or equal to 0.5 and mapped to a EMA (equation 4) if the particle had a value less than 0.5.

$$EMA_t = EMA_{t-1} + \alpha * (P_t - EMA_{t-1}) \tag{4}$$

where, α is determined by the equation $\alpha = 2/(\mathrm{N} + 1)$ where N is the size of the window and P_t is the value of the underlying financial asset at time t. Each

Table 1. The parameters that the PSO algorithm optimized. MA stands for moving average and the type of MA could be simple or exponential.

Symbol	Description
N	The size of the moving window for calculating the MA.
K_u	The value for calculating the upper band.
K_l	The value for calculating the lower band.
T	The type of MA to use.

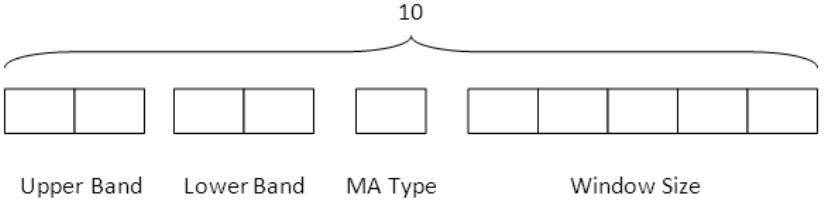


Fig. 2. The mapping of the 10-dimensional position vector to the BB parameters for each particle in the swarm

experiment was performed with 50 iterations as the stopping criterion and each particle velocity and its corresponding position in the 10-dimensional space was updated with equations 5 and 6 respectively.

$$v_{i,j} = \omega * v_{i,j} + C_1 R_1 * (local_{best\,i,j} - x_{i,j}) + C_2 R_2 * (global_{best\,j} - x_{i,j}) \quad (5)$$

Here $v_{i,j}$ is the velocity of j^{th} dimension of the i^{th} particle, c_1 and c_2 determine the influence on a particular particle by its optimal position previously visited and the optimal position obtained by the swarm as a whole, r_1 and r_2 are uniform random numbers between 0 and 1, and ω is an inertia term (see [6]) chosen between 0 and 1.

$$x_{i,j} = x_{i,j} + v_{i,j} \quad (6)$$

Here $x_{i,j}$ is the position of the j^{th} dimension of the i^{th} particle in the swarm. To encourage exploration and limit the speed with which the swarm would converge, a maximum velocity was chosen for each dimension dependent on its range of feasible mappings. In table 2 the range and maximum velocity for each parameter is displayed.

4 Fitness Functions

The overall goal of the experiment is to determine the optimal parameters for maximizing profits, it would seem obvious that training the swarm with a fitness function based on profit would be the most appropriate. However, other

Table 2. The range of feasible values for each parameter and its corresponding maximum velocity for navigating the solution space

Parameter	Range	Max Velocity
Upper Band	{4,4}	0.75
Lower Band	{4,4}	0.75
MA Type	{0,1}	0.10
Window Size	{0,500}	20

literature, Moody et al. [5], found that optimal performance was arrived at with fitness functions which have a risk to reward payoff.

The four fitness functions are described below and each contains a component for transaction costs. The profit fitness function rewarded particles which generated the highest level of profit after accounting for transaction costs without any regard for how risky the trading model was. The fitness function, shown in equation 7, is a sum over all trades (T) taken by the model:

$$fitness_i = \sum_{i=1}^{T} capital_t * \frac{(P_{1,t} - P_{0,t})}{P_{0,t}} - (\tau * capital_t) \tag{7}$$

where $fitness_i$ is the fitness of the i^{th} particle in the swarm, τ represents the transaction costs, and P_0 and P_1 are the entering and exiting price for the underlying asset. The profit for each trade is the rate of return multiplied by the capital invested minus the transaction cost which is also a function of the amount of capital invested. The Sharpe and Sortino ratios are commonly used metrics for evaluating how efficient a trading model is with the additional risk it is exposed to. In both cases a higher value indicates a more efficient use of risk. The Sharpe Ratio is shown in equation 8:

$$S = \frac{E[R - R_f]}{\sqrt{var[R - R_f]}} \tag{8}$$

where R is the return on the asset and R_f is a risk-free rate and E is the expected returns operator. The only difference in the above equation for the Sortino ratio is the denominator: where the Sharpe uses all returns from the trading model, the Sortino only considers negative returns, thus models are allowed to have a higher variance, as long as the returns are positive. Using these equations to derive the fitness function will allow for models which efficiently use risk to be assigned higher fitness with the intention that these models will perform better in the out-of-sample test periods. The fitness functions employed thus far are path dependent and therefore can be sensitive to initial conditions. To help assess the limitations of the prior approaches we introduce a fourth fitness function (equation 9) that assesses how often the trading model produces a positive gain in relation to the total number of positions it takes in the market.

$$fitness_i = \frac{\#returns > 0}{\#returns > 0 \ + \ \#returns < 0} \tag{9}$$

5 Data Description and Experiment Design

The dataset contains daily market values for the DJIA spanning 20 years from Jan 1990 to Dec 2009, where a split of 10 years for training and 5 years for testing yielded 2 separate but not fully independent test periods for the index. The four PSO models are trained separately on each of the datasets and then the optimized parameters are tested on the 5 year out-of-sample data. For comparison's sake the models are compared to a buy-and-hold approach and the default settings for the BBs using an SMA and an EMA. The default settings for SMA and EMA are a 20 day sliding window and a value of 2 for k. The transaction costs are considered to be 0.5% of the capital invested and the risk-free rate (R_f) is a constant 2%.

6 Experimental Results

The results from training are displayed in table 3; the cumulative returns are based on a $100 initial investment.

Table 3. Training results for each model and time period

	Return $		Return %		# of trades	
Model	90-99	95-04	90-99	95-04	90-99	95-04
Profit	101.17	105.26	0.011	0.052	44	42
Sharpe	-3.54	4.62	-1.03	-0.953	42	42
Sortino	-1.896	109.86	-1.01	0.098	44	48
Accuracy	22.29	32.84	-0.777	-0.671	44	44
EMA	52.62	58.63	-0.473	-0.413	112	94
SMA	30.00	43.22	-0.699	-0.567	166	164
Buy-Holds	409.12	280.91	-	-	-	-

From the training results we can see that models created from any of the BB parameter settings were inferior to the buy and hold approach. However the PSO model utilizing the profitability fitness function (PSO_{profit}) was the top performer amongst the BB models. The PSO models all produced a trading activity of around 40 transactions; a manifestation of the 40 trade minimum requirement that was imposed. Such a constraint was added because of the poor performance of the models with the most profitable being those which traded very little or not at all. The test results, reported in table 4, are more competitive between the buy and hold approach and the BB trading models, where the buy and hold approach was outperformed by PSO_{profit} between 2000 and 2004. In the second testing period the buy and hold approach was the most profitable but PSO_{profit} was a close second. The PSO models tended to trade less often than with the default settings, reducing the burden of transaction costs.

Table 4. Testing results for each model and time period

	Return $		Return %		# of trades	
Model	00-04	05-09	00-04	05-09	00-04	05-09
Profit	101.44	94.31	0.014	-0.056	16	18
Sharpe	87.37	89.94	-0.126	-0.100	20	20
Sortino	61.50	68.45	-0.384	-0.315	36	16
Accuracy	78.04	81.30	-0.219	-0.186	32	34
EMA	92.02	84.70	-0.079	-0.152	48	64
SMA	88.61	87.65	-0.113	-0.123	82	90
Buy-Holds	94.94	97.19	-	-	-	-

7 Discussion

If the primary goal of using Bollinger bands is to maximize profit then it seems obvious that a fitness function which optimizes on this metric would be the most reasonable to implement. As discussed, there is supporting literature that has found that using a risk-adjusted fitness function delivers superior out-of-sample returns. In this study this conclusion is not consistent and though these studies involve financial time-series with similar objectives (maximizing profits), the actual task of the algorithms is different and therefore may require different fitness functions. With regards to the Sharpe and Sortino ratios, they were unable to promote movement of the swarm from unfit regions of the solution space because of the calculations themselves. When all models are producing negative returns the less fit particles are being chosen to lead the swarm. From equation 8 we see that if we have two models which produce the same negative return, the

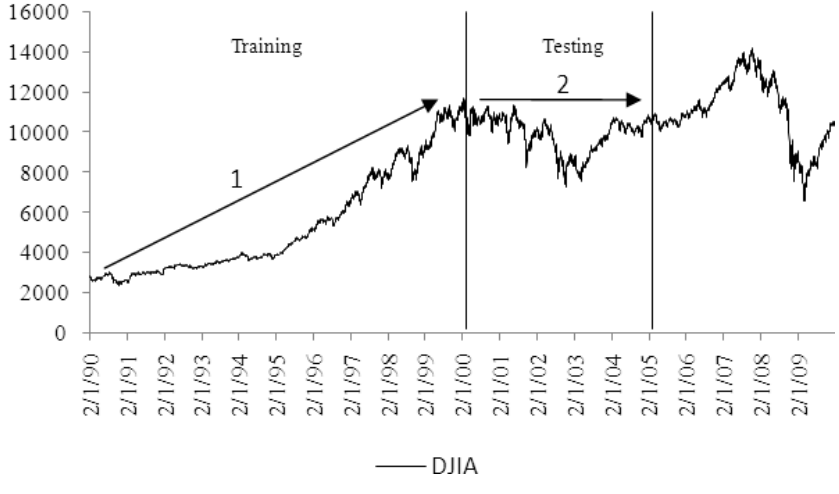


Fig. 3. A plot of the DJIA over both sets of training and testing data. Arrow 1 signifies the upward trend of the market during the first training phase and arrow 2 signifies the sideways moving market in the first testing period.

one which is more volatile and therefore less favorable will actually be ranked higher. The accuracy fitness function was generally outperformed by the other PSO methods, possibly from the lack of consideration for magnitude in the returns when assigning fitness. Although the optimization was able to outperform the default settings in the training and testing periods, it is apparent that when the market is trending, the BBs are unable to capture the excess profits. In one testing period a PSO model was able to outperform the buy and hold approach but this was in the wake of a market contraction. In figure 3 we have a graph of the DJIA over the training and testing periods.

8 Conclusions and Future Work

The main objective of this study was to investigate the short-comings of Bollinger Bands as technical indicators and if a swarm intelligence approach to optimizing its parameters would assist its effectiveness. This included an analysis of 4 different fitness functions which are commonly used in financial modeling and a comparison to the traditional use of the indicator and a market portfolio. The results to date have shown that by fine-tuning the parameters over a training period, with an appropriate fitness function, the particle swarm optimization algorithm is able to find an optimal set of parameters and that depending on the market environment is able to outperform a buy and hold approach. Future work will include expanding the study to other market indices, as well as, changing the fitness ranking of the Sharpe and Sortino ratios. Finally, some trading rules such as stop losses could be imposed to cap how much a trade is able to lose which could assist the accuracy fitness function as it would not be as susceptible to large losses during fitness assignment.

References

1. Lee, J.S., Lee, S., Chang, S., Ahn, B.H.: A comparison of ga and pso for excess return evaluation in stock markets. In: Mira, J., Álvarez, J.R. (eds.) IWINAC 2005, Part II. LNCS, vol. 3562, pp. 221–230. Springer, Heidelberg (2005)
2. Lento, C., Gradojevic, N.: The profitability of technical trading rules: a combined signal approach. Journal of Applied Business Research 23(1), 13–27 (2007)
3. Lento, C., Gradojevic, N., Wright, C.: Investment information content in Bollinger Bands? Applied Financial Economics Letters 3(4), 263–267 (2007)
4. Leung, J., Chong, T.: An empirical comparison of moving average envelopes and Bollinger Bands. Applied Economics Letters 10(6), 339–341 (2003)
5. Moody, J., Wu, L., Liao, Y., Saffell, M.: Performance functions and reinforcement learning for trading systems and portfolios. Applied Financial Economics Letters 17, 441–470 (1998)
6. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, IEEE World Congress on Computional Intelligence, pp. 69–73 (1998)
7. Williams, O.: Empirical Optimization of Bollinger Bands for Profitability. Master's thesis, Simon Fraser University (2006)

Protein Structure Prediction in Lattice Models with Particle Swarm Optimization

Andrei Băutu and Henri Luchian

Faculty of Computer Science, "Al. I. Cuza" University, Iasi, Romania
abautu@anmb.ro, hluchian@uaic.ro

Abstract. The protein structure prediction problem consists in finding good computational algorithms for prediction of protein native states. This paper applies the Particle Swarm Optimization (PSO) algorithm to predict the tertiary structure of proteins in lattice models. We propose a novel discrete PSO variant designed for lattice-based protein folding models. We present three lattice based models and two folding encodings, which are tested in different combinations on six proteins. The results indicate that the new algorithm performs very efficient and finds very good proteins conformations.

1 Introduction

The protein structure prediction (PSP) problem is part of the larger protein folding problem and consists in finding good computational algorithms for predicting protein native states based on amino acid sequences. Current research focuses on two main directions: prediction based on existing databases of protein foldings and similarities between proteins, and prediction using physics laws, without derived knowledge [11]. Our research lays on the latter path.

This paper applies the Particle Swarm Optimization (PSO) algorithm to predict the tertiary structure of proteins in lattice models based on their primary structure. We propose a novel approach, which uses a discrete PSO variant designed for lattice-based protein folding models. Despite the simplicity of these types of models, the protein folding problem is still very hard.

The following section briefly presents the terminology and goals of the PSP problem. It also presents three frequently used lattice models. Section 3 briefly describes the binary PSO algorithm. Section 4 presents an extension of the binary PSO to the PSP problem in lattice models and introduces a new discrete PSO, Roulette PSO, which "boroughs" the roulette wheel selection of Genetic Algorithms. Section 5 presents the experimental results obtained for six proteins with various lattice models and folding encodings. The last section contains some remarks about current status and future work directions.

2 Protein Structure Prediction Problem

Proteins are the most important molecules found in living cells because they perform a vast range of indispensable roles: speeding up chemical reactions, regulating cell activity, protecting cells, transporting elements, etc. From a chemical

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 512–519, 2010.

point of view, they are organic compounds created from sequences of amino acids (AAs). Each AA has a central $\alpha-$carbon which is linked to three attachments similar to all AAs and a residual attachment, which differentiates various AAs. Ribosomes sequence AAs according to DNA instructions into the primary structure of the protein. To carry out its tasks, the protein evolves by linking AAs with hydrogen bonds (the secondary structure), folding the resulting polypeptide chain into three-dimensional structures (the tertiary structure) and coupling together multiple chains (the quaternary structure).

The tertiary structure of the protein is called the native state and represents the energetic ground state of the protein. Various simplified models for the protein structure exist: the Toy model, the Functional Model Protein, the Hydrophobic-Polar model, etc. Despite the simplicity of these models, the protein folding problem is still very hard. Designing efficient algorithms for the PSP problem is an important research area of computational biology.

The full complexity of the folding processes that take place in proteins are yet to be unveiled. Even if they were known, detailed all atom simulations of such complex processes would not be possible in modern computers, so simpler models were developed. The dominant driving force for protein folding is the hydrophobic force [9]. With respect to this, AAs are hydrophobic or hydrophilic. In water environments, hydrophobic AAs tend to cluster in order to minimize contact with water, while hydrophilic AAs do not avoid contact with water.

Lattice models focus only on the general principles of the folding process. Sites available for AAs are located in 2D square or 3D cubic lattices, the distances between adjacent AAs are fixed and equal, and the bond angles can be ± 90 or 180 degrees. The general energy function for folding in lattice models is

$$E = \sum_{i<j} \epsilon_{a_i a_j} \delta(p_i, p_j) \ , \tag{1}$$

where a_i and a_j denote the types of i^{th} and j^{th} AAs, which are located on the lattice in sites p_i and, respectively, p_j. The $\epsilon_{a_i a_j}$ constants depend on the model and they control the type of interaction between different types of AAs. The values of the δ function depend on the folding and they control the contribution of pair interactions to the total system energy.

The HP model focuses on short-range contacts of hydrophobic AAs. The energy of the folding decreases only for hydrophobic not-linked AAs which are next to each other in the lattice. Therefore, the interaction constants in (1) are $\epsilon_{\text{HH}} = -1$, $\epsilon_{\text{PP}} = 0$, $\epsilon_{\text{PH}} = 0$, and $\epsilon_{\text{HP}} = 0$. The attenuation function δ is 1 if AAs i and j are not linked and occupy neighboring sites, and it is 0 otherwise:

$$\delta(p_i, p_j) = \begin{cases} 1, \text{ if } |i-j| > 1 \text{ and } \|p_i - p_j\|_1 = 1 \\ 0, \text{ otherwise} \end{cases} \ , \tag{2}$$

where $\|\cdot\|_1$ denotes the Manhattan distance. The HP model is a focus of research in computational biology and statistical physics [20].

The functional model protein (FMP) uses the same attenuation function as HP, but it includes repulsive interactions. Hydrophobic not-linked AAs next to

each other decrease the system energy by 2 ($\epsilon_{\mathrm{HH}} = -2$). The other types of interactions (i.e. H-P, P-H, P-P) increase the system energy by 1 ($\epsilon_{\mathrm{PP}} = 1$, $\epsilon_{\mathrm{PH}} = 1$, and $\epsilon_{\mathrm{HP}} = 1$).

The HP and FMP models use the concept of Free Energy [10] to measure the energy of foldings. They consider only interactions of AAs in neighbor sites of the lattice. For the rest of AAs, it makes no distinction if they are located close or far apart from each other. Therefore, the energy landscape for most proteins is discrete, step-wise and with large plateaus. To address this problem, [5] defines a Global Energy (GE) function which uses the distance between AAs, smoothing the energy landscape. The GE model is a variant of the HP model, which uses the inverse of the Euclidean distance ($\|\cdot\|_2$) as the attenuation function:

$$\delta(p_i, p_j) = \|p_i - p_j\|_2^{-1} \ . \tag{3}$$

The PSP problem is $\mathcal{NP}$-hard in many protein folding models (including the simple HP model [6]).

3 Particle Swarm Optimization

PSO is a meta-heuristic based on the principles of swarm intelligence. It uses a set of potential solutions (called particle swarm) to solve optimization problems. It was proposed for continuous optimization problems in [12] and later adapted for binary [14] and discrete domains. The objective function describes the optimization problem and defines the problem landscape in terms of solutions quality. Particles fly in the problem landscape, searching for high quality solutions. The particles communicate with each other in a collaborative search effort. The driving force of PSO is the collective swarm intelligence, which proved very successful in tackling various difficult problems [1].

The search process starts with a swarm of particles randomly scattered in the search space. On each iteration, particles adjust their velocity v and position p using information gathered by themselves or received from their neighbors. The classical equation for velocity update combines individual and social learning sources with the current state of the particle (v' and p'):

$$v = \omega v' + R_1(p_{\mathrm{p}} - p') + R_2(p_{\mathrm{g}} - p') \ . \tag{4}$$

The ω parameter controls the inertia of the particle. The p_{p} and p_{g} are the best positions found by the particle and its neighbors. The R_1 and R_2 are random Uniformly distributed variables that weight the learning sources. The amplitude of the velocity vector is clamped by a $v_{\max}$ parameter, which prevents the "explosion" of the swarm [7].

The position of each particle is updated based on the new velocity. In the case of binary PSO, the position vector $p \in \{0,1\}^n$ contains the responses of the particle for the n binary queries of the problem. The probability that the particle will answer 1 to a particular query i depends on the velocity of the particle in that query plane (v_i):

$$p = \begin{cases} 1, \text{ if } R_3 < (1 + \exp(-v))^{-1} \\ 0, \text{ otherwise} \end{cases}, \tag{5}$$

where R_3 is a random Uniformly distributed variable in $[0, 1)$.

Before the next iteration, the fitness of each particle is computed based on the problem solution encoded in the position, p_{p} and p_{g} are updated, and the best solution found so far is stored for presenting it as the algorithm output. For more details on PSO we refer the reader to [8].

4 PSO for Protein Structure Prediction

Real-valued PSO was applied successfully on the PSP problem with various off-lattice models, like the Toy model or HP model with ECEPP energy function [17,18,22]. Although the search space is larger, the off-lattice models give PSO more freedom to exploring it, more informative feedback, and provide a smooth energy landscape. The successes of real-valued PSO for off-lattice protein folding models are encouraging, proving once again the power of the particle swarm paradigm.

Many types of optimization techniques provide good results on lattice models [3,21,13,16]. In [2], a discrete PSO for PSP on lattice models is proposed. It proves successful in finding low energy foldings for proteins with various sizes. Compared to a genetic algorithm (GA), PSO performed better because it ignored local minima foldings and identified more native states than the GA. This paper continues that research, by testing the PSO algorithm on more lattice models.

4.1 Folding Representation

Encoding the folding of a protein in a lattice-based model can be done in many ways. The common approach is to encode absolute or relative folding directions. In absolute encoding, a folding code for each AA sets the next folding direction in the context of a fix coordinate system. In relative encoding, a folding code for each AA sets the next folding direction in the context of a coordinate system relative to the last direction. For example, in the case of a 2D lattice, the absolute folding instructions can be *Up, Down, Left, Right* and the relative ones can be *Left, Right, Ahead* (see Fig. 1). Each of the two codings has advantages and disadvantages: the relative encoding has less folding codes and instructions (i.e. smaller search space); the absolute encoding is more stable to changes. [15] recommends relative encodings for genetic algorithms, while [2] reports improved performance for PSO with the absolute encoding.

4.2 Particle Search Space

The number of possible codes for each folding operation (m) depends on the lattice (i.e. 2D or 3D) and the folding instruction set (i.e. absolute or relative). The PSO particles need to be able to explore a search space that represents all valid conformations. For example, when using the absolute encoding in a 2D

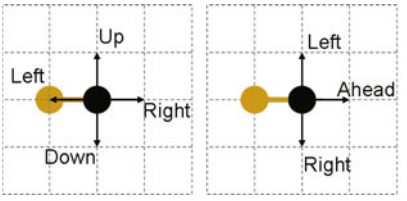


Fig. 1. Absolute (left) and relative (right) folding instructions on a 2D lattice

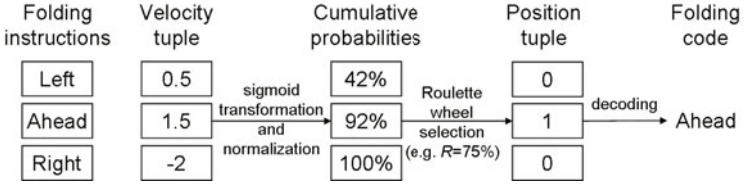


Fig. 2. Schematic representation of the decision process of RPSO

lattice, each element of the position vector should be able to hold 4 possible values. Moreover, the particle must be able to swing directly between any of these values, similar to the oscillations used by real-valued PSO particles to find solutions for real-valued optimization problems [19].

The PSO algorithm from [2] assigns a different integer number for each folding code, starting from 0, and use their base 2 representations in particle positions. The representation mappings used for 2D lattice are down=00, up=01, left=10, right=11 for absolute encoding, and ahead=00 or 01, left=10, right=11 for relative encoding. This simple representation permits the use of the binary PSO without modifications. If the protein has n AAs, then the particle and velocity vectors have $\lceil \log_2 m \rceil (n-1)$ components for absolute encoding and, respectively, $\lceil \log_2 m \rceil (n-2)$ components for relative encodings. The disadvantage of this representation is that a folding code requires multiple independent stochastic decisions (using (5)).

Inspired by binary PSO and roulette wheel selection of GAs, we designed a PSO algorithm to address this disadvantage, called Roulette PSO (RPSO). RPSO tracks independently the probability of each folding instruction to be selected for a folding code. For each folding code, the corresponding velocity component is a tuple of m real value elements. The i^{th} element of this velocity tuple encodes the probability that the i^{th} folding instruction should be used for this code. The position component is a tuple of m binary values with only one bit set to 1 (and the rest are 0). If the i^{th} element of the position is set to 1 then the i^{th} folding instruction is used in the code. The velocity is updated with (4) meaning that a bad folding instruction yields from its selection probability in favor of a good folding instruction. Instead of (5), the position is updated using roulette wheel selection from GAs: the probability of each folding instruction is computed from the velocity tuple; a random number $R \in [0, 1)$ is produced; the

Table 1. Proteins used in experiments

Code	Protein string	Size (n)	E^*
P1	3H 1P 1H 5P 1H	11	-2
P2	1H 1P 1H 2P 2H 1P 2H 1P 1H 1P 2H 2P 1H 1P 1H	20	-9
P3	2H 2P 1H 2P 1H 2P 1H 2P 1H 2P 1H 2P 1H 2P 2H	24	-9
P4	2P 1H 2P 2H 4P 2H 4P 2H 4P 2H	25	-8
P5	3P 2H 2P 2H 5P 7H 2P 2H 4P 2H 2P 1H 2P	36	-14
P6	P H 2P H P 3H P 2H P 5H	18	-9

position bit for the instruction with the smallest cumulative probability larger than R is set to 1. This process is represented in Fig. 2 for a single folding code.

5 Experimental Results

We compared a set of experiments using proteins presented in Tab. 1. The first column identifies the protein code referred in the following discussions. The second column contains the protein HP string. The fourth column contains the minimum energy of the protein in the HP model on 2D lattice [20,4].

For each protein, we tested the binary PSO from [2] and RPSO with absolute and relative encodings on the lattice models presented in Sec. 2. For each experimental setup 15 independent runs were performed, with 100 iterations each. The PSO algorithms used 300 particles connected in a star topology, learning factors (R_1 and R_2) are randoms variables from U[0,1), $\omega = 1$, and $v_{\max} = 5$.

Figure 4 contains boxplots for the folding energies of each protein in the three lattice models (due to its simplicity, all algorithms performed equally good on protein P1, so it was excluded from the plot). The X-values denote the tested algorithm variant: A-B = absolute encoding with binary PSO, A-R = absolute encoding with RPSO, R-B = relative encoding with binary PSO, R-R = relative encoding with RPSO. The statistics recorded during the runs indicate that RPSO with absolute encoding (A-R) had the lowest energy mean in most case. The next best option in many cases is binary PSO with absolute encoding (A-B). The worst results were obtained by binary PSO with relative encoding (R-B). None of the algorithms ever found the perfect folding for problem P5 in the HP model, but RPSO with absolute and relative encodings found highly compact foldings (see Fig. 3) which are only 1 energy-unit away from the perfect solution. RPSO requires 50% to 85% more run time than binary PSO on the

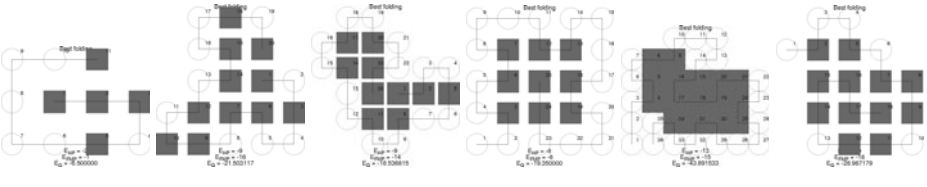


Fig. 3. Foldings examples for each protein found by the RPSO algorithm

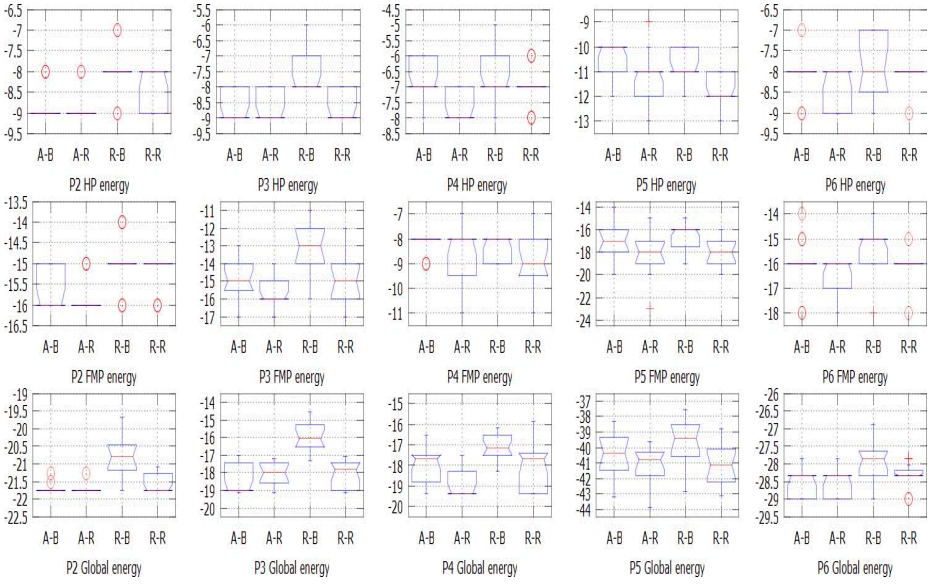


Fig. 4. Boxplots of folding energies

same encodings. There is no significant difference in run times between the two encodings. With respect to the iteration in which the solution was found, there is no significant difference between A-R, A-B and R-R. However, R-B frequently required more iterations to locate its solutions and they had higher energy.

6 Conclusions

This paper presents a new discrete PSO for the protein structure prediction problem on lattice models. Experimental results are very encouraging as the conformations obtained by the algorithm are comparable in quality with state-of-the-art results. However, the perfect conformation of one of the proteins (P5) eluded the algorithms' search process proving that there is still room for improvement (e.g. hybridization with local search algorithms).

Acknowledgments. This paper is supported by the NatComp PN II-11-028/2007 project.

References

1. Băutu, A., Băutu, E.: Particle Swarms in Statistical Physics, pp. 77–88. Intech Publishing (2009)
2. Băutu, A., Luchian, H.: Protein structure prediction in the 2D HP model using Particle Swarm Optimization. In: Proc. of 7th Int. Conf. on Numerical Methods and Applications (to appear)

3. Benítez, C.M.V., Lopes, H.S.: A parallel genetic algorithm for protein folding prediction using the 3D-HP side chain model. In: Proc. of CEC 2009, pp. 1297–1304. IEEE Press, Piscataway (2009)
4. Bennett, A.J., Johnston, R.L., Turpin, E., He, J.Q.: Analysis of an immune algorithm for protein structure prediction. Informatica (2008)
5. Berenboym, I., Avigal, M.: Genetic algorithms with local search optimization for protein structure prediction problem. In: Proc. of the 10th annual Conf. on Genetic and evolutionary computation, pp. 1097–1098. ACM, New York (2008)
6. Berger, B., Leighton, T.: Protein folding in the hydrophobic-hydrophilic (HP) is NP-complete. In: Proc. of RECOMB 1998, pp. 30–39. ACM, New York (1998)
7. van den Bergh, F., Engelbrecht, A.: A study of particle swarm optimization particle trajectories. Information Sciences 176(8), 937–971 (2006)
8. Clerc, M.: Particle Swarm Optimization. Hermes Science, London (2006)
9. Dill, K.A.: Dominant forces in protein folding. Biochemistry 29(31), 7133–7155 (1990)
10. Dill, K.A., Ozkan, S.B., Shell, M.S., Weikl, T.R.: The protein folding problem. Annual Review of Biophysics 37(1), 289–316 (2008)
11. Dill, K.A., Ozkan, S.B., Weikl, T.R., Chodera, J.D., Voelz, V.A.: The protein folding problem: when will it be solved? Current Opinion in Structural Biology 17(3), 342–346 (2007)
12. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proc. of the Sixth Int. Symp. on Micro Machine and Human Science, pp. 39–43 (1995)
13. Kapsokalivas, L., Gan, X., Albrecht, A., Steinhöfel, K.: Population-based local search for protein folding simulation in the MJ energy model and cubic lattices. Computational Biology and Chemistry 33(4), 283–294 (2009)
14. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: Proc. of the World Multiconf. on Systemics, Cybernetics and Informatics, vol. 5, pp. 4104–4109. IEEE Press, Piscataway (1997)
15. Krasnogor, N., Hart, W.E., Smith, J., Pelta, D.A.: Protein structure prediction with evolutionary algorithms. In: Proc. of the Genetic and Evo. Comp. Conf., vol. 2, pp. 1596–1601. Morgan Kaufmann, Orlando (1999)
16. Lin, C.J., Hsieh, M.H.: An efficient hybrid Taguchi-genetic algorithm for protein folding simulation. Expert Systems with Applications 36(10), 12446–12453 (2009)
17. Liu, J., Wang, L., He, L., Shi, F.: Analysis of Toy Model for Protein Folding Based on PSO Algorithm. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 636–645. Springer, Heidelberg (2005)
18. Pérez-Hernández, L.G., Rodríguez-Vázquez, K., Garduno-Juárez, R.: Parallel PSO applied to the protein folding problem. In: Proc. of the 11th Annual Conf. on Genetic and evolutionary computation, pp. 1791–1792. ACM, New York (2009)
19. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intelligence 1, 33–57 (2007)
20. Santana, R., Larranaga, P., Lozano, J.A.: Protein folding in simplified models with EDA. IEEE Transactions on Evolutionary Computation 12(4), 418–438 (2008)
21. Thachuk, C., Shmygelska, A., Hoos, H.: A replica exchange Monte Carlo algorithm for protein folding in the HP model. BMC Bioinformatics 2008(1), 342 (2007)
22. Zhu, H., Pu, C., Lin, X., Gu, J., Zhang, S., Su, M.: Protein structure prediction with EPSO in Toy model. In: Proc. of ICINIS 2009, pp. 673–676. IEEE Computer Society, Washington (2009)

Short and Robust Communication Paths in Dynamic Wireless Networks

Yoann Pigné[1] and Frédéric Guinand[2]

[1] SnT, University of Luxembourg, Luxembourg
yoann.pigne@uni.lu
[2] LITIS, University of Le Havre, France
frederic.guinand@univ-lehavre.fr

Abstract. We consider the problem of finding and maintaining communication paths in wireless mobile ad hoc networks (MANET). We consider this problem as a bi-objective problem when trying to minimize both the length of the constructed paths and the number link reconnections. We propose two centralized algorithms that help analyse the problem from a dynamic graph point of view. These algorithms give lower bounds for our proposed decentralized ant-based algorithm that constructs and maintains such paths in a MANET.

1 Introduction

Mobile ad hoc networks (MANETs) define communication networks composed of mobile devices or stations able to communicate together with wireless media. When communicating in a peer-to-peer way, they do not rely on any infrastructure. The resulting communication network is decentralized. The volatility and the mobility are two major characteristics of these networks. The mobility comes with the nature of these stations that are usually small devices handheld by their owner or embedded in mobile vehicles. The volatility illustrates the idea that stations are not always turned on. They can be standing by and then reactivate to reappear in their neighborhood.

Stations volatility is the results of poor capacity batteries while radio communication need a lot of energy. As a result, energy saving is a key issue in the design of applications dedicated to MANETs that have to be careful with communication overheads. Two stations willing to communicate need to exchange preliminary information to set up the link. This initialization is called here, for seek of simplicity, "synchronization". Moreover, discovery times may also be considered (for instance, the Service Discovery Protocol in Bluetooth may be too slow for mobile networks). Synchronization processes need a lot of energy achieve and the number of new communication links may be minimized.

MANETs as well as classical networks need communication services like data routing. Yet, classical networks protocols for routing reveal to be useless in infrastructure-less communication systems. MANETs need dedicated algorithms for the routing of communication. A large panel of algorithms are proposed to

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 520–527, 2010.

perform routing in MANETs, taking into account different constraints such as the overhead of transmitted data [6], the quality of the communication links [4] or the geographical motion of the stations [5].

Here we address the problem of finding and maintaining communication paths between couples of stations in a MANET. We consider the problem of extra energy consumption that occurs when mobility and volatility force new synchronization processes. Our purpose is to minimize the number of these new synchronization processes by selecting communication paths that are less likely to disconnect. Such selected paths are called robust paths. In this paper we focus on a global analysis of this property in dynamic graphs with centralized algorithms (Sect. 2). This analysis gives clues to manage paths in a decentralized environment. Section 3 focuses on a decentralized ant-based approach that deals with the MANET issues. Simulations results are shown Sect. 4.

2 Global Analysis

The aim of this paper is to study the possibilities of creating and maintaining short and robust paths in mobile ad hoc networks. This global analysis aims at giving lower bounds to compare with the decentralized algorithm. Simulated ad hoc networks produce produce communication networks that can be considered as dynamic graphs where stations are nodes and effective communication links between stations are edges. These dynamic graphs are replayed and analyzed centrally. We propose some measurement for the problems we consider. First the length of paths is considered. Then for the robustness we suggest the *renewal rate* measurement: Let s be a subset of edges and nodes in the dynamic graph. s is said to be a structure. s is observed at to different dates—t_1 and t_2—during the evolution of the dynamic graph. Let S_{t_1} be the structure s at time t_1 and S_{t_2} the same structure at time t_2. Since the graph is a dynamic one, the set of edges and nodes that constitute s may change between t_1 and t_2. The renewal rate $t_r(S_{t_1}, S_{t_2})$ is the number C of changes (addition and removal of elements) in s between t_1 and t_2 divided by the cardinality of S_{t_1}: $RR(S_{t_1}, S_{t_2}) = \frac{C}{|S_{t_1}|}$.

We propose two centralized algorithms to produce lower bounds for the next decentralized approach.

Shortest Paths Minimal Set (SPMS). At each step of a dynamic graph one compute the minimal list of shortest paths between a given source and a destination. From the first step $i = 0$ the set of shortest paths is computed and is said to be the reference structure. Iteratively steps of the dynamic graph are applied. For each step a new shortest path structure is computed. If a path exists in this intersection structure then it becomes the new reference structure. The number of shortest paths cannot be larger in the new reference structure than in the older one since the new one is included in the old one. On the contrary, if the intersection between the new structure and the reference structure does not have a shortest path, then it means that the reference shortest path has been lost in this new step (t). Anyway we know that this structure is valid until the step $t-1$ so it is stored as the good one until the step $t-1$ and the number of shortest

path structures in the solution set is increased by one. For this new step the new shortest path structure becomes the new reference and the process can go one with another step. At the end of this process, only the minimal set of shortest path structures is stored.

Robust Structures Minimal Set (RSMS). The constraint of shortest path is relaxed and only the existence of a path between a pair of nodes is considered. It is interesting in our case to look for the minimal set of paths that may link two nodes during the evolution of the graph. The algorithm constructs with the same behavior as above (algorithm SPMS) the minimal set of paths that link two nodes in a dynamic graph. Since several paths may link the nodes at any moment we consider structures that links them. We look for paths or structures that last as long as possible without changing—robust structures.

RSMS does not have optimal renewal rate; however experiments show that it is a good lower bound for the heuristic method presented next section.

Proposed algorithms give interesting measures for the deferred analysis of communication networks. These algorithms give lower bounds for the two objectives that we consider—short and robust paths. For now, after this global analysis, online constraints may be considered with a decentralized approach.

3 Ant-Based Construction and Maintaining of Robust Paths

We now focus on applied constraints that lead to the conception of dedicated and decentralized algorithms in the field of mobile ad hoc networks. We consider networks where the mobility of stations may highly impact on the kind of service that can be delivered. We choose a realistic simulation mobility that corresponds to the mobility of pedestrians in an urban environment. Here, communication devices are handheld devices that may usually be shut down, so we consider a volatility model for the stations.

3.1 Description of the Model

The idea here is to design a decentralized algorithm that is able to handle the changes in the environment. Natural ant colonies have the ability to construct and maintain short path in changing environments. Even the loss of some ants is supported. The number of entities needed here is a key issue. Several previous works are related like AntNet [3] and Ant-Based Control (ABC) [8] in wired networks, and more precisely *AntHocNet* [2], DAR (Distributed Ant Routing) [7], or HOPNET [9] in wireless networks.

In this model, ants do not construct solutions to a given problem. They are just influenced by pheromone tails when making journeys in a dynamic network and also lay pheromone down the environment. On the wireless network field, the environment is the set of stations and pheromone trails are updated in stations local routing tables.

This algorithm uses the same local exploration formulas as defined in ACOs. For an ant k located on node i of a graph at date t, the choice for a next node j is made locally according the probability:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in nbs(i) \backslash path(k)} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}} & \text{if } j \in nbs(i) \backslash path(k) \\ 0 & \text{else} \end{cases} \quad (1)$$

where:

- $\tau_{ij}(t)$ is the quantity of pheromone on edge (i, j) at step t;
- η_{ij} is the visibility on edge (i, j)—explained below;
- $nbs(i)$ is the set of neighboring nodes of i;
- $path(k)$ is the list of nodes that constitute the path already done by ant k;
- $neighbors(i) \backslash path(k)$ is the set of neighbors of i deprived of the node that are those node that are in the path of ant k;
- α and β are two parameters that relatively change the weight of pheromone trails in comparison to the visibility.

The two local information that are able to lead ants in the graph are the pheromone trails and the visibility. We want this information to be related to the two objectives we consider—short and robust paths. Pheromone trails are known to help ants construct short paths. The visibility is meant to be a local estimation of the quality of the path being constructed. Here the visibility is meant to give indications on the robustness of the paths being constructed. We need a local estimation of this robustness that is a global measurement, like the renewal rate (Sect. 2).

We propose to use a local heuristic based on the age of the edges to estimate their robustness. Let the number of appearances be the number of times a given edge appears in the dynamic graph. Let the overall age of this edge be the sum of the ranges of time it exists in the dynamic graph. We call volatility the ratio between the *number of appearances* of an edge and its *overall age*: $volatility = \frac{number\ of\ appearances}{overall\ age}$.

3.2 Algorithm

Ants move in the graph according to two modes—a forward mode when looking form the destination and a backward mode when the destination is reached. This model is common to several algorithms just like DAR, ARA or AntHocNet.

Forward mode. The ant moves from node to node looking for the destination. It stores the constructed path. Next node is chosen according to Eq. (1).

Backward mode. When an ant in forward mode reaches the destination node, then it goes backward to the nest using the path created during the forward

mode. It updates the pheromone trails. Similarly to ACOs the reinforcement of pheromone trails is made proportionally to the length of the constructed path.

Evaporation. As defined in ACO, the parameter ρ rules the evaporation rate of pheromone to be applied on each edge. With the constatnt Q, the quantity of pheromone (τ_{ij}) on edge (i, j) is: $\tau_{ij} = (1 - \rho)(\tau_{ij} - Q) + Q = \tau_{ij} - \rho(\tau_{ij} - Q)$.

3.3 Memory of the Pheromone Trails

Similarly to the volatility measurement that is computed from the logging of the appearances and disappearances of edges by nodes, the values of pheromone of an edge may be stored even when an edge disappears. This memory allows to give back to an edge its pheromone trail when it reappears. The idea is that if an edge has a high quantity of pheromone then it may a good point to quickly give it back its attractiveness when it reappears.

This heuristic may be discussed. A part of the simulations results are dedicated to the analysis of this mechanism and show its efficiency under some hypothesis. Experiments show that this memory of pheromone trails reveals to be very efficient for environments with high volatility.

4 Simulations and Results

We want to focus here on the simulation of realistic networks composed of handheld devices such as PDAs, cell phones or laptops that are able to communicate with *IEEE 802.11b* radio chips. We consider urban environments such as mall centers or streets or hallways. Tow different scenarios with different characteristics are proposed.

The first scenario called "hallway" is distinguished by a very high mobility of the stations but with no volatility—the stations move fast and their radio device is open. The environment is a hallway and the mobility is constrained by this hallway such that stations may only move from one part of the hall to the other in the two possible directions. The speed of the stations is correlated to the speed of walking people. All the parameters that identify the scenario are displayed Table 1.

The second scenario called "volatility" highlights a weak mobility and a strong volatility for its stations, as show parameter values in Table 1. Stations evolve in an open environment with a Random Way Point [1] mobility model.

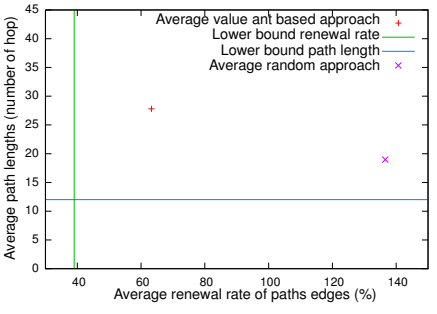
Simulations are based on a discrete time evolution by the simulator Madhoc (`litis.univ-lehavre.fr/~hogie/madhoc/`). Videos of these simulations are also available at `litis.univ-lehavre.fr/~pigne/these/`.

4.1 Multi-Objective Optimization

Two objectives are considered in this problem: paths lengths and renewal rate of paths edges, both being minimized. Figure 1 gives a general overview of all the results obtained with the algorithm for simulations with the "volatility"

Table 1. Simulation parameters for scenarios "hallway" and "volatility"

scenario	"hallway"	"volatility"
Mobility	two directions	RWP
Environment	closed (hallway)	open
Area	$500m$ long hallway	$1km^2$
Stations	150	400
Station celerity	1.4 to $2m/s$	0 to $1m/s$
Volatility	no volatility	40% of stations
Radio radius	$20m$	$40m$



- Horizontal line: average shortest path on each pair of source destination nodes.
- Vertical line: lower bound for the renewal rate given by the RSMS algorithm (Sect. 2).
- "+": average value for the solutions generated by the ant based approach.
- "×": average value for the solutions generated by a random centralized search algorithm.

Fig. 1. General solutions layout in the multi-objective solution space

scenario. Data here are average values for different parameter sets, 8 different pairs of source and destination, and 10 different random seeds. Four values are presented in the multi-objective space of solutions where x-axe gives the average renewal rate of the edges of a path and the y-axe gives the length of a path. The observation is that the produced results are less efficient than the random search in terms of path length. However the average renewal rate of the edges of the solutions generated by the ant-based approach is far more efficient than the random method.

4.2 Parameters and Results

Metaheuristics are usually criticized for their important set of parameters to be tuned. Our proposed model also suffers criticism; however dependencies between parameters allow simplifications in the system. Relative importance parameters α and β (Eq. (1)) are relative one on the other. α is set to 1 and the evolution of β is observed. Then we consider ρ, the evaporation rate of the pheromone trails. It may strongly impact the algorithm's behavior changing the attractiveness of the paths. Other parameters are tuned experimentally as shown in the table of Fig. 2. This figure shows the results obtained by our proposed algorithm with the simulation of the "volatility" scenario. Each point on the figure is the average value of the paths constructed by the algorithm when ran on the network with 8 different pairs of source/target nodes and 6 random seeds (48 different runs). From one point to another only the values of the parameters β and ρ differ.

Parameter	Description	Value
Q	initial pheromone	1
ρ	evaporation rate	variable
$nbAnts$	number of ants	40
k	node jumps per step	100
α	pheromone strength	1
β	visibility strength	variable
TTL	ant max lifetime	100

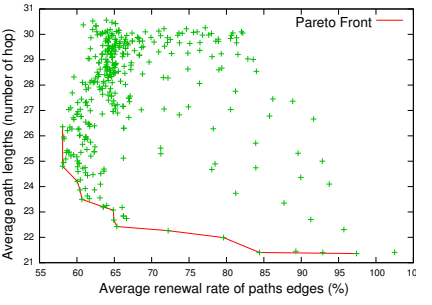
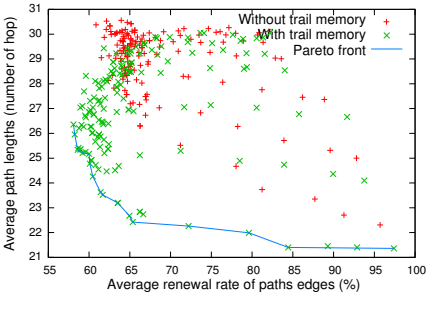


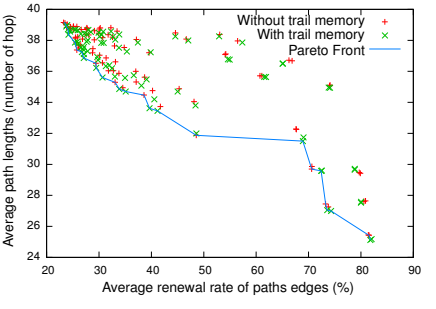
Fig. 2. Parameters and solutions for the "Volatility" scenario

4.3 Analysis of the Pheromone Trail Memory Heuristic

The proposed heuristic is to use a memory to recall pheromone values on edges that reappear after having disappeared. The hypothesis is that an edge with a strongly loaded pheromone trail is probably an important edge that has to get his attractiveness back as quickly as possible. In the "volatility" scenario edges that disappear have a high probability to reappear since the mobility is low. In this case the heuristic seams appropriated. Figure 3(a) shows the undeniable superiority of this approach. On the opposite the memory heuristic is not appropriated to the "hallway" scenario where edges that disappear have nearly no chance to reappear. Figure 3(b) illustrates this.



(a) "Volatility" scenario



(b) "Hallway" scenario

Fig. 3. Pheromone trail memory heuristic on the two scenarios

5 Conclusion

The purpose we were given here was to address the problem of finding and maintaining robust and short communication paths in dynamic and decentralized environments such as mobile ad hoc networks. This problem leads us to

consider the problem as a two objective problem. Swarm intelligence and especially ant-based approaches have shown efficiency in evolving and decentralized environments. We proposed such an ant-based models that only depends on local heuristics and is able to adapt to the changes in the environment.

A study of the problem with centralized algorithms has been performed and allows identifying key issues in the problem. This analysis helped the formulation and the evaluation of the decentralized ant based algorithm.

Two original propositions show up in this field. First the use of the local volatility measurement of the edges allows—as shown by the simulations—to lower the renewal rate of the paths edges constructed by the ants. This means in terms of mobile wireless that the overhead communication needed to reconnect communication links is lowered. The second original proposition is that a local heuristic computed by the stations of a mobile network leads to store pheromone trails of communication links that disappear so as to give them back there pheromone value if they ever reappear. This last heuristic shows it efficiency when stations are volatile and communication links have chances to disappear and then reappear.

References

1. Broch, J., Maltz, D.A., Johnson, D.B., Hu, Y.C., Jetcheva, J.: A performance comparison of multi-hop wireless ad hoc network routing protocols. In: MobiCom 1998, pp. 85–97. ACM, New York (1998)
2. Di Caro, G., Ducatelle, F., Gambardella, L.M.: Anthocnet: an ant-based hybrid routing algorithm for mobile ad hoc networks. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 461–470. Springer, Heidelberg (2004)
3. Di Caro, G., Dorigo, M.: Antnet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research 9, 317–365 (1998)
4. Gaertner, G., ONuallain, E., Butterly, A., Singh, K., Cahill, V.: 802.11 Link Quality and Its Prediction - An Experimental Study. In: Niemegeers, I.G.M.M., de Groot, S.H. (eds.) PWC 2004. LNCS, vol. 3260, pp. 147–163. Springer, Heidelberg (2004)
5. Ko, Y.B., Vaidya, N.H.: Location-aided routing (lar) in mobile ad hoc networks. In: MobiCom 1998: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, pp. 66–75. ACM Press, New York (1998)
6. Perkins, C., Royer, E.: Ad-hoc on-demand distance vector routing. In: WMCSA 1999: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications, p. 90. IEEE Computer Society, Washington (1999)
7. Rosati, L., Berioli, M., Reali, G.: On ant routing algorithms in ad hoc networks with critical connectivity. Ad Hoc Networks 6(6), 827–859 (2008)
8. Schoonderwoerd, R., Holland, O., Bruten, J., Rothkrantz, L.: Ant-based load balancing in telecommunications networks. Adaptive Behavior 5(2), 169 (1997)
9. Wang, J., Osagie, E., Thulasiraman, P., Thulasiram, R.K.: Hopnet: A hybrid ant colony optimization routing algorithm for mobile ad hoc network. Ad Hoc Networks 7(4), 690–705 (2009)

The ACO Encoding

Alberto Moraglio, Fernando E.B. Otero, and Colin G. Johnson

School of Computing and Centre for Reasoning,
University of Kent, Canterbury, UK
{A.Moraglio,F.E.B.Otero,C.G.Johnson}@kent.ac.uk

Abstract. Ant Colony Optimization (ACO) differs substantially from other meta-heuristics such as Evolutionary Algorithms (EA). Two of its distinctive features are: (i) it is constructive rather than based on iterative improvements, and (ii) it employs problem knowledge in the construction process via the heuristic function, which is essential for its success. In this paper, we introduce the *ACO encoding*, which is a *self-contained algorithmic component* that can be readily used to make available these two particular features of ACO to *any* search algorithm for continuous spaces based on iterative improvements to solve combinatorial optimization problems.

1 Introduction

Hybrid meta-heuristics that combine skilfully elements of two or more meta-heuristics are often superior to the original meta-heuristics when considered in isolation [2] as it is the case, for example, of Memetic Algorithms [6], which combine Evolutionary Algorithms [1] with Local Search, and more recent incarnations of Ant Colony Optimization [4] which also embed Local Search as a standard sub-component. The success of hybrid meta-heuristics is rooted in the combination of the complementary strengths of their compounding meta-heuristics [2].

Leaving aside the compelling ant foraging behavior metaphor that inspired the original design of ACO, from an algorithmic point of view, Ant Colony Optimization differs greatly from most meta-heuristics. Perhaps the two most characterizing algorithmic features of ACO are: (i) it is constructive rather than based on iterative improvements; and (ii) beside the feedback obtained by the evaluation of complete solutions, common to all other meta-heuristics, the heuristic function provides extra problem knowledge fed directly in the construction process.

One weakness of ACO is that the pheromone update rules are not guaranteed to find always the optimal pheromone levels that lead to the construction of the best solution. This is because the update rules make small incremental changes to the pheromone levels—which can be seen as a form of hill-climbing in the pheromone space—that may lead the pheromone levels to converge to a point which is only locally optimal. It would be interesting, therefore, to replace the search done in the pheromone space by the update rules with more global types

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 528–535, 2010.

of search algorithms, such as Evolutionary Algorithms, which may be less prone to get stuck in local optima.

Beside the standard hybridization of ACO with Local Search, there are a number of works on hybrid ACO algorithms, see for example [5,12,11]. In this paper, rather than introducing one more hybrid ACO algorithm, we intend to present a *self-contained algorithmic component* obtained by distilling a distinctive feature of ACO behind its success—namely the use of heuristic information in the solution construction—in a way that it can be used as an off-the-shelf component in any search algorithm for continuous spaces to solve combinatorial problems. The new algorithmic component will be termed the *ACO Encoding*. This component was derived derandomizing the ACO construction mechanism and interpreting it as a decoding procedure for solutions of combinatorial problems represented as real vectors formed by pheromone levels, as illustrated in the next section. This component has advantages from two different perspectives: (i) it can be seen as endowing (a de-randomized variant of) ACO with a more global search in the pheromone space than that using the standard pheromone update rule, and, at the same time, (ii) it feeds extra problem knowledge via the heuristic function to the base search algorithm (e.g., an evolutionary algorithm) using the ACO encoding, which normally does not have such knowledge at its disposal.

2 The ACO Encoding

Search algorithms are normally characterized by:

1. the search space S defining the set of candidate solutions to the problem at hand P (i.e., the set of feasible solutions).
2. the underlying representation R of the candidate solutions (e.g., real vectors, permutations, binary strings) on which the search operators O are acting on.
3. the search strategy A that the search algorithm employs to search the given search space (e.g., local search, population-based search, annealing search).

Whereas perhaps ACO is not normally looked at this way, it is insightful to ask what its search space, its solution representation and its search strategy are.

Let us consider a basic ACO without local search for a specific problem, e.g., TSP (i.e., P=TSP). The problem being addressed is clearly a combinatorial problem. A standard solution representation for the TSP are permutations (i.e., R=permutations), which naturally encode the order of the cities to be visited by the traveling salesperson (i.e., S=permutations (*genotypes*) encoding cities tours (*phenotypes*)). As ACO constructs TSP solutions encoded using permutations, it may be argued that the construction procedure employed by ACO corresponds to the "ACO search operator" in the space of permutations (i.e., O=construction procedure). However, this way of aligning ACO to the algorithmic framework outlined above is partially unsatisfactory because there are no elements in ACO that *directly* correspond to search operators acting on such a representation. As a consequence, the search strategy A of ACO cannot be explicitly characterized on the permutation space.

Let us now consider an alternative way of looking at ACO, attempting at equating the search space of ACO with the set of all pheromone values. In this case, TSP solutions are constructed using the information stored in the pheromone values (i.e., S=pheromones vectors (genotypes) encoding cities tours (phenotypes)). Importantly, the pheromone updates rule acting on pheromone levels can be interpreted as search operators acting on such a representation (i.e., R=pheromones vectors and O=update rules). The solution construction can be then interpreted as a *growth function* that maps pheromone levels (genotypes) to permutations (intermediate phenotypes), which are in turn mapped to cities tours (phenotypes). Interestingly, in ACO the growth function maps a continuous space to a combinatorial space, hence ACO, seen in this way, searches a combinatorial space indirectly by searching a continuous space. Can the pheromone space, therefore, be considered as the search space searched by ACO? The answer is negative as an essential property of a solution representation is missing when we consider the pheromone levels as representation. This is the ability of the genotype to identify uniquely a phenotype. This is essential, as in lack of it, it would not be possible to determine the fitness of the genotype unambiguously, and as a consequence, the optimal genotype would not be well-defined.

If we are willing to allow for a change of the ACO solution construction procedure, we could indeed identify the pheromone space with the search space of ACO. There are two aspects of the ACO solutions construction procedure that are problematic when it is interpreted as a growth function: (i) normally, the ACO construction procedure returns more than one solution (one for each ant on the graph); and (ii) it is a probabilistic procedure which does not always construct the same solutions from the same pheromone levels. The first issue can be simply solved by using always a single ant. The second issue can be solved by derandomizing the construction procedure. This can be done by placing the ant on a fixed initial node (i.e., the nest) and by de-randomizing its decisions about which node to go next by always choosing the alternative with highest probability, i.e. *by treating transition probabilities as priorities* and returning deterministically only the most probable solution. The priorities used in the de-randomized construction are obtained with the traditional ACO transition probability formula which combines heuristic information and pheromone levels. Since the heuristic information does not change in the course of the search, these priorities are uniquely determined by the pheromone levels on the construction graph, and consequently, given the same pheromone levels, the TSP tour built by the ant is unique. So, this construction procedure is a well-defined growth function.

The above derandomized variation of ACO is a degenerate form of ACO, which, as it is, does not perform any useful search. This is because it produces deterministically a single solution from the current pheromone levels, and when the pheromone update rule is applied, the pheromone values corresponding to that solution are reinforced, hence, fixating the search to that single solution. However, the interesting aspect of this derandomized variation of ACO is that it forms a conceptual bridge between the traditional ACO and other types of metaheuristics as this algorithm can now be characterized in terms of search space,

solution representation and search operators. Seen in this algorithmic framework, the ACO representation of solutions (pheromone levels) together with *the ACO encoding* (deterministic solution construction procedure) can be naturally decoupled from the specific ACO search (pheromone update rule). *This, interestingly, allows us to use the ACO encoding as a self-contained algorithmic component in combination with any search algorithm for continuous optimization.* So, for example, we could use it in combination with a standard Evolutionary Algorithm based on a real vector representation. Each real vector (individual in the population) corresponds to a combination of pheromone levels. The fitness of that individual would be the fitness of the solution constructed by the ant using those pheromone levels. Then, the evolutionary algorithm would proceed as it normally does by selecting above average individuals, recombining and mutating them with standard search operators, to produce the next population of individuals.

The conceptual link between ACO and other meta-heuristics via the ACO Encoding is interesting, as the same real values assume two quite distinct meanings when they are understood in the two different contexts: as pheromone values in an ACO construction graph, and as solution representation in the genotype of an individual. Let us consider more closely this difference for the specific case of TSP. For this problem, in the ACO practice, the pheromone is normally stored on edges rather than on nodes of the construction graph, because its function is understood being that of "modifier" of the heuristic information, which is associated with the edges of the construction graph (since it is based on the distance between the cities). On the other hand, if we look at pheromone values as a solution representation for an EA, as a rule of thumb, we want to have the most compact representation that can represent any TSP solution, because larger spaces take normally more time to be searched. This would require to put pheromone on the nodes of the construction graph, for a total of n pheromone locations, rather than pheromone on edges which would amount to a total of n^2 pheromone locations, hence giving rise to a much larger search space to search. Note that putting pheromone on nodes is enough to represent any solution of the TSP, as when the heuristic information is set to zero, the pheromone levels on the nodes, in fact, specify the order of the cities to include in the salesperson tour by their ranks. This is known in the literature as the random-keys encoding for permutations [8]. Therefore, there seems to be a very interesting inter-play between heuristic information injected in the ACO Encoding and its redundancy.

The idea of injecting heuristic information in the decoding procedure from genotype to phenotype is not new in the ACO Encoding, as it has been used before, for example, in encoding procedures for scheduling problems using the genotypes to guide the choice of scheduling policies [7]. However, the ACO Encoding makes it possible to use heuristic information in a much more standardized way and for a much larger class of problems, borrowing from the large and rapidly increasing library of previously solved problems using ACO.

3 Experiments and Discussion

Let us consider the ACO Encoding included as a component in a evolutionary algorithm for real-vectors. The ACO Encoding raises a number of interesting questions. In the previous section, we decoupled the ACO Encoding (solution representation and construction) from the ACO search in pheromone space (pheromone update rule). The first natural question is: given the same ACO Encoding of the problem at hand, is the more global search done by an EA better than the search done by the ACO pheromone update rule? A second interesting aspect of the ACO encoding is that it allows the inclusion of heuristic information in the EA. So, a second question is: since the use of this information is essential in ACO to obtain good performance, is the heuristic information of any help to the EA? A third question relates with the dual interpretation of the real-vector as pheromone in a construction graph and as genotype to represent a solution, as discussed in the previous section. So, we ask which of the two following options is better: (i) locating pheromone on the edges of the construction graph, which from an ACO perspective it is meaningful for the TSP, or alternatively (ii) locating pheromone on the nodes of the construction graph, which makes sense from an EA perspective as it gives rise to a smaller search space to search.

In the following, we present three sets of experiments using 10 TSP instances from the TSPLIB to give preliminary answers to the above questions. As our aim is to compare directly algorithmic components rather than reaching state-of-the-art performances, we use the ACO Encoding with a very simple evolutionary algorithm for real-vectors (a generational scheme with discrete uniform crossover, creep mutation and tournament selection). Furthermore, we do not include local search neither in ACO nor in the EA and, for fairness, we give the same number of fitness evaluations (number of solutions constructed and evaluated) to each of the algorithms in the comparison.

We have selected two well-known ACO algorithms, namely ant colony system (ACS) [3] and $\mathcal{MAX}$-$\mathcal{MIN}$ ant system ($\mathcal{MM}$AS) [9,10]. For the ACO algorithms, the following default parameter settings from the literature are used: $\beta = 2$, $\alpha = 1$, $m = n$ (where m is the number of ants and n is the number of cities) and $\rho = 0.98$ (where ρ is the evaporation rate). In the case of our EA using the proposed ACO Encoding, we have used three variations: EA-ACO, EA-ACO-b0 (EA-ACO without heuristic information) and EA-ACO-n (EA-ACO with pheromone on the nodes). We have performed a systematic (but coarse) parameter tuning in order to determine a suitable combination of values, leaving $\alpha = 1$ and $\beta = 2$ parameters fixed—the only exception is that in EA-ACO-b0, β is set to 0 since it does not use heuristic information. For the remaining parameters, we have tested the following values: $\mathit{population\_size} = \{n \cdot 10, \boldsymbol{n \cdot 10}\}$, $\mathit{crossover\_rate} = \{0.5, \mathbf{1.0}\}$, $\mathit{mutation\_rate} = \{1/n, \mathbf{2/n}\}$ and $\mathit{tournament\_size} = \{\mathbf{2}, 4\}$. The values in bold represent the best combination of values out of the 16 possible combinations and the ones used in our experiments. Although the parameter tuning did not show significant differences between the combinations of values, the EA seems to be more sensitive to the *tournament_size* parameter, where the combinations using the smaller value 2 perform better than the ones

Table 1. Computational results for symmetric (upper section) and asymmetric (lower section) instances from the TSPLIB. The number following the instance name corresponds to the number of cities. The column '*opt*' indicates the known optimal tour length for each instance. A value in the remaining columns represents the average tour length of the best tour found by the correspondent algorithm over 25 runs (*average*±*standard deviation*); the value closer to the optimal tour length is shown in bold.

instance	*opt*	ACS	$\mathcal{MM}$AS	EA-ACO	EA-ACO-b0	EA-ACO-n
berlin52	7542	7868.9±192.7	**7562.8±57.4**	7849.8±68.7	9585.1±460.2	7886.0±31.6
eil51	426	468.5±11.8	**428.7±1.6**	430.6±4.1	504.8±13.3	439.9±2.9
kroA100	21282	28497.8±562.6	**21492.8±145.2**	21852.8±148.2	69800.5±2424.2	22306.5±131.0
st70	675	806.6±15.2	**681.5±4.5**	715.1±8.1	1206.9±48.6	712.9±11.6
br17	39	**39.0±0.0**	**39.0±0.0**	41.7±6.4	39.0±0.2	39.7±3.4
ft70	38673	45072.3±320.0	39586.2±331.2	39732.4±200.3	44474.2±689.3	**39370.8±154.4**
p43	5620	5630.9±1.3	**5629.3±0.9**	5652.6±6.8	5673.5±15.5	5638.5±6.4
ry48p	14422	16147.9±335.9	14598.3±70.6	**14534.6±46.6**	17252.4±631.5	14721.4±145.2

using the greater value 4. The stopping criteria for all algorithms was set to $14 \cdot 10^4$ fitness evaluations. Therefore, even though the algorithms use a different number of ants (ACO algorithms) and population size (EA algorithms), the comparison is based on the total number of candidate solutions evaluated, which will be the same for all algorithms.

Table 1 presents the computational results. For each of the algorithms, we report the average tour length of the best tour found in 25 independent runs of the algorithms. The value closer to the optimal tour length is shown in bold in Table 1. The results show that generally $\mathcal{MM}$AS achieves the best performance. The only exception are the 'ft70' and 'ry48p' asymmetric instances, where the EA-ACO and EA-ACO-n achieve best performance. It is interesting to note that EA-ACO and EA-ACO-n outperform ACS, except on 'br17', where ACS found the optimal solution, and on 'p43'. However, EA-ACO-b0 performs poorly compared to the other algorithms, except for the 'br17' instance, where it outperforms the other two EA variations. In terms of computational time, both ACS and $\mathcal{MM}$AS require on average 27 seconds to complete the $14 \cdot 10^4$ fitness evaluations, EA-ACO-n is a factor of 2.6 slower than ACS and $\mathcal{MM}$AS, while EA-ACO and EA-ACO-n are a factor of 110 slower than ACS and $\mathcal{MM}$AS.

Let us now consider the questions that were posed at the beginning of the section in the light of the experimental results obtained. Given the same encoding, is the EA search (EA-ACO) better than the ACO search (ACS and $\mathcal{MM}$AS) by means of the update rule? From the experiments, it seems that the EA search reaches better solutions than the simple ACS, but it does not as well as the more sophisticated $\mathcal{MM}$AS. Note that one of the major differences between ACS and $\mathcal{MM}$AS is the ability of the latter to change adaptively the learning rate to prevent getting stuck in local optima in the phenotype space. This seems to be consistent with our initial hypothesis that the EA search is more global than

the simple search done by ACS. However, the adaptive search of $\mathcal{MM}$AS seems to be very effective. The EA used in this work is a very simple one. It would be, therefore, interesting in future work to consider EA with an adaptive type of mutation and compare it with $\mathcal{MM}$AS. In terms of CPU time, the ACO systems are much more efficient than the EA-ACO. The critical component seems to be the crossover operator that takes quadratic time with respect with the number of cities. However, this problem may be overcome by using a parameter set for the EA that requires to apply crossover only a limited number of times.

A second question was about the impact of the heuristic information injected in the EA using the ACO Encoding. From the experiments, it is evident that the heuristic information in EA-ACO clearly allows it to outperform the same algorithm when this information is not available (EA-ACO-b0). This is a very interesting result as it shows that the ACO Encoding is indeed able to inject heuristic information in a EA in a way that can be usefully employed to reach better performance. More generally, the ACO Encoding may open up at the possibility to inject heuristic information in any search algorithm operating on a continuous representation and systematically boost its performance.

A third question we put forward was about whether it is better to locate the pheromone on edges (as in EA-ACO) or on the nodes (as in EA-ACO-n). In terms of efficiency, which is, CPU time needed, EA-ACO-n is much more efficient than the EA-ACO, as the time required by the crossover operator is now linear with the number of cities in a solution. In terms of performance, the pheromone on edges seem to be more advantageous than the pheromone on nodes, but not of much. This is interesting as it shows something very counter-intuitive from an EA point-of-view: a heavily redundant representation corresponding to a much larger than necessary search space to search seems to lead to a better performance than a more compact one. It is also interesting to note that EA-ACO-n, which incorporates heuristic information, is both superior to and much more efficient than EA-ACO-b0, which has pheromone on edges and it does not incorporate heuristic information. Therefore, EA-ACO-n seems to be an interesting trade-off in terms of performance, efficiency and use of heuristic information.

Finally, we can consider the experimental results above as representing a test of whether the effectiveness of ACO is due to the *encoding/representation* or to the *search* algorithm. The results tentatively suggest that the *encoding/representation* is at the core of the power of ACO, as replacing one search algorithm by another does not compromise the effectiveness of the search. Naturally, more experimentation across problem domains is needed to provide stronger support for this argument. This constitutes an important piece of future work.

4 Conclusions

This paper has introduced the ACO Encoding, an algorithmic component obtained by decoupling the encoding used in ACO from the search component, so that the encoding can be used with other search techniques. We have demonstrated some basic experiments on TSP instances, consisting of combining this encoding with a very simple EA search, showing a similar rate of success to a

good ACO variant, and presented an analysis of these results. In future work, we will use the ACO encoding with state-of-the-art algorithms for continuous optimization, such as differential evolution and self-adaptive evolution strategies, in the attempt to reach top performance on hard combinatorial optimization problems.

References

1. Bäck, T., Fogel, D.B., Michalewicz, T. (eds.): Evolutionary Computation 1: Basic Algorithms and Operators. Institute of Physics Publishing (2000)
2. Blum, C., Blesa Aguilera, M.J., Roli, A., Sampels, M. (eds.): Hybrid Metaheuristics: An Emerging Approach to Optimization. Springer, Heidelberg (2008)
3. Dorigo, M., Gambardella, L.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
4. Dorigo, M., Sützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
5. Lee, Z.J., Su, S.F., Chuang, C.C., Liu, K.H.: Genetic algorithm with ant colony optimization (ga-aco) for multiple sequence alignment. Applied Soft Computing 8(1), 55–78 (2008)
6. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Tech. rep., Caltech Concurrent Computation Program (1989)
7. Runwei Cheng, M.G., Tsujimura, Y.: A tutorial survey of job-shop scheduling problems using genetic algorithmsi. representation. Computers and Industrial Engineering 30(4), 983–997 (1996)
8. Snydera, L.V., Daskin, M.S.: A random-key genetic algorithm for the generalized traveling salesman problem. European Journal of Operational Research 171(1), 38–53 (2006)
9. Stützle, T., Hoos, H.: Improvements on the Ant System: Introducing $\mathcal{MAX}$-$\mathcal{MIN}$ ant system. In: Proc. Int. Conf. Artificial Neural Networks and Genetic Algorithms (1997)
10. Stützle, T., Hoos, H.: $\mathcal{MAX}$-$\mathcal{MIN}$ ant system. Future Generation Computer Systems 16(8), 889–914 (2000)
11. Wong, K.Y., See, P.C.: A hybrid ant colony optimization algorithm for solving facility layout problems formulated as quadratic assignment problems. Engineering Computations: Int. J. for Computer-Aided Engineering 27(1), 117–128 (2010)
12. Xiong, W., Wang, C.: A hybrid improved ant colony optimization and random forests feature selection method for microarray data. In: International Conference on Networked Computing and Advanced Information Management (2009)

The Complexity of Grid Coverage by Swarm Robotics

Yaniv Altshuler[1,2] and Alfred M. Bruckstein[1]

[1] Computer Science Department, Technion, Haifa, Israel
{yanival,freddy}@cs.technion.ac.il
[2] Deutsche Telekom Labs, Ben Gurion University, Beer Sheva, Israel

Abstract. In this paper we discuss the task of efficiently using ant-like robotic agents for covering a connected region on the $\mathbf{Z}^2$ grid, whose shape and size are unknown in advance, and which expands at a given rate. This is done using myopic robots, with no ability to directly communicate with each other, where each robot is equipped with only $O(1)$ memory. We show that regardless of the algorithm used, and the robots' hardware and software specifications, the minimal number of robots required in order to enable such coverage is $\Omega(\sqrt{n})$ (where n is the initial size of the connected region). In addition, we show that when the region expands at a sufficiently slow rate, a team of $\Theta(\sqrt{n})$ robots could cover it in at most $O(n^2 \ln n)$ time. Regarding the coverage of non-expanding regions in the grid, we improve the current best known result of $O(n^2)$ by demonstrating an algorithm of worse case completion time of $O(\frac{1}{k}n^{1.5} + n)$, and faster for shapes of perimeter length which is shorter than $O(n)$.

1 Introduction

In this paper we examine a problem in which a group of ant-like robotic agents must cover an unknown region in the grid, that possibly expands over time. This problem is also strongly related to the problem of distributed search after mobile and evading target(s) [4,3] or the problems discussed under the names of "Lions and Men" pursuits [6]. We analyze such issues using the results presented in [11, 1,2], concerning the *Cooperative Cleaners* problem, a problem that assumes a regular grid of connected 'tiles', part of which are 'dirty', the 'dirty' tiles forming a connected region of the grid. On this dirty grid region several agents move, each having the ability to 'clean' the place it is located in. In the dynamic variant of this problem a deterministic evolution of the environment in assumed, simulating a spreading *contamination* (or spreading *fire*).

First, we discuss the collaborative coverage of static grids. We demonstrate that the best completion time known to date ($O(n^2)$, achievable for example using the $LRTA*$ search algorithm) can be improved to guarantee grid coverage in $O(\frac{1}{k}n^{1.5} + n)$ time. Later, we discuss the coverage of expanding domains. We show that using any conceivable algorithm, and using as sophisticated and potent robotic agents as possible, the minimal number of robots below which covering such a region is impossible equals $\Omega(\sqrt{n})$. We then show that when

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 536–543, 2010.

the region expands sufficiently slow (specifically, every $O(\frac{c_0}{\gamma_1})$ time steps, where c_0 is the circumference of the region and where γ_1 is a geometric property of the region, ranging between $O(1)$ and $O(\ln n)$), a group of $\Theta(\sqrt{n})$ robots can successfully cover the region. Furthermore, we demonstrate that in this case a cover time of $O(n^2 \ln n)$ is guaranteed.

2 Related Work

A search for analytic results concerning the completion time of ant-robots covering an area in the grid revealed only a handful of works. The main result in this regard is that of [7], where a swarm of ant-like robots is used to repeatedly cover an unknown area, using a real time search method called *node counting*, using integer markers that are placed on the graph's nodes. The cover time of teams of ant robots that use node counting is shown in [8,9] to be $t_k(n) = O(n^{\sqrt{n}})$, when $t_k(n)$ denotes the cover time of a region of size n using k robots. Another algorithm to be mentioned in this scope is *LRTA**, whose multi-robotics variant is shown in [8] to guarantee cover time of undirected connected graphs in polynomial time. Specifically, on grids it guarantee coverage in $O(n^2)$ time.

Vertex-Ant-Walk, a variant of the node counting algorithm presented in [12], is shown to achieve a coverage time of $O(n\delta_G)$, where δ_G is the graph's diameter, implying a worse-case completion time of $O(n^2)$ in grids. This work is based on a previous work in which a cover time of $O(n^2\delta_G)$ was demonstrated [10].

An algorithm named *Exploration as Graph Construction*, providing a coverage of degree bounded graphs in $O(n^2)$ time, can be found in [5]. Here a group of limited ant robots explore an unknown graph using special "markers".

We next show that the problem of collaborative coverage in static grids can be completed in $O(\frac{1}{k}n^{1.5} + n)$ time and that collaborative coverage of dynamic grids can be achieved in $O(n^2 \ln n)$.

3 The Dynamic Cooperative Cleaners Problem

Following is a short summary of the *Cooperative Cleaners* problem, as appears in [11] (static variant) and [1,2] (dynamic variant). Assuming a discrete time, let the undirected graph $G(V, E)$ denote a two dimensional integer grid $\mathbf{Z}^2$, whose vertices (or "*tiles*") have a binary property called '*contamination*'. Let $cont_t(v)$ state the contamination state of the tile v at time t, taking the values "*on*" or "*off*". Let F_t be the contaminated sub-graph of G at time t, and let F_0 be simply connected. Let a group of k robots that can move on the grid G (moving from a tile to its neighbor in one time step) be placed at time t_0 on F_0, at point $p_0 \in F_t$. Each robot is equipped with a sensor capable of telling the status of all tiles in the digital sphere of diameter 7, which surrounds the robot. Each robot is equipped with a memory of size $O(1)$ bits. When a robot moves to a tile v, it has the possibility of cleaning this tile (i.e. causing $cont(v)$ to become *off*. The robots do not have any prior knowledge of the shape or size of the sub-graph F_0.Every d time steps, the contamination spreads. That is, if $t = nd$ for some positive

integer n, then $\forall v \in F_t \ \forall u \in 4-Neighbors(v)$, $cont_{t+1}(u) = on$. The robots' goal is to clean G by eliminating the contamination entirely. It is important to note that no central control is allowed, no communication is allowed, and that the system is fully decentralized.

A Survey of Previous Results. The cooperative cleaners problem was previously studied in [11,1,2]. A cleaning algorithm called **SWEEP** was proposed (used by a decentralized group of simple mobile robots, for exploring and cleaning an unknown "contaminated" sub-grid F, expanding every d time steps) and its performance analyzed. The **SWEEP** algorithm is based in a constant traversal of the contaminated region, preserving the connectivity of the region while cleaning all *non critical points* — points which when cleaned disconnect the contaminated region. Following are several results that we later use. Note that *cleaning* a region is equivalent to *covering* it, as the number of uncovered tiles is upper bounded by the number of remaining contaminated ones.

Result 1 ***(Cleaning a Non-expanding Contamination).*** *The time it takes for a group of K robots using the* ***SWEEP*** *algorithm to clean a region F of the grid is at most:*

$$t_{static} \triangleq \frac{8(|\partial F_0| - 1) \cdot (W(F_0) + k)}{k} + 2k$$

$W(F)$ denotes the depth of the region F (the shortest path from some internal point in F to its boundary, for the internal point whose shortest path is the longest) and ∂F denotes the boundary of F, defined via $\partial F = \{v \mid v \in F \ \wedge \ 8 - Neighbors(v) \ \cap \ (G \ \setminus \ F) \ \neq \ \emptyset\}$.

Result 2 ***(Universal Lower Bound on Contaminated Area).*** *Using any cleaning algorithm, the area at time t of a contaminated region that expands every d time steps can be recursively lower bounded, as follows:*

$$S_{t+d} \geq S_t - d \cdot k + \left\lfloor 2\sqrt{2 \cdot (S_t - d \cdot k) - 1} \right\rfloor$$

Here S_t denotes the area of the contaminated region at time t (such that $S_0 = n$).

Result 3 ***(Upper Bound on Cleaning Time for SWEEP on Expanding Domains).*** *For a group of k robot using the* ***SWEEP*** *algorithm to clean a region F on the grid, that expands every d time steps, the time it takes the robots to clean F is at most d multiplied by the minimal positive value of the following two numbers:*

$$\frac{(A_4 - A_1 A_3) \pm \sqrt{(A_1 A_3 - A_4)^2 - 4A_3(A_2 - A_1 - A_1 A_4)}}{2A_3}$$

where:

$$A_1 = \frac{c_0 + 2 - \gamma_2}{4} \quad , \quad A_2 = \frac{c_0 + 2 + \gamma_2}{4} \quad , \quad A_3 = \frac{8 \cdot \gamma_2}{d \cdot k} \quad , \quad A_4 = \gamma_1 - \frac{\gamma_2 \cdot \gamma}{d} \quad ,$$

$$\gamma_1 = \psi(1+A_2) - \psi(1+A_1) \quad , \quad \gamma_2 = \sqrt{(c_0+2)^2 - 8S_0 + 8} \quad ,$$
$$\gamma = \frac{8(k+W(F_0))}{k} - \frac{d-2k}{|\partial F_0| - 1}$$

Here c_0 is the circumference of the initial region F_0, and where $\psi(x)$ is the *Digamma* function — the logarithmic derivative of the *Gamma function*.

4 Grid Coverage — Analysis

We first present the cover time of a group of robots operating in non-expanding domains, using the **SWEEP** algorithm.

Theorem 1. *Given a connected region of $S_0 = n$ tiles and perimeter c_0, then k ant-like robots can cover it using $O\left(\frac{1}{k}S_0^{1.5} + S_0\right)$ time.*

Proof. Since $|\partial F_0| = \Theta(c_0)$, and $W(F_0) = O(\sqrt{S_0})$, recalling Result 1 we see that:

$$t_k(n) = t_{static}(k) = O\left(\frac{1}{k}\sqrt{S_0}\cdot c_0 + c_0 + k\right)$$

As $c_0 = O(S_0)$ and as for practical reasons we assume that $k < n$ this equals:

$$t_k(n) = t_{static}(k) = O\left(\frac{1}{k}S_0^{1.5} + S_0\right)$$

We now examine the problem of covering expanding domains. The lower bound for the number of robots required for completing is as follows.

Theorem 2. *Given a region of size $S_0 \geq 3$ tiles, expanding every d time steps, then a team of less than $\frac{\sqrt{S_0}}{d}$ robots cannot clean the region, regardless of the algorithm used.*

Proof. Recalling Result 2, and by assigning $k = \frac{\sqrt{S_0}}{d}$ we can see that:

$$\Delta S_t = S_{t+d} - S_t \geq \left\lfloor 2\sqrt{2\cdot(S_t - \sqrt{S_0}) - 1} \right\rfloor - \sqrt{S_0}$$

For any $S_0 \geq 3$, we see that $\Delta S_0 > 0$. In addition, for every $S_0 \geq 3$ we can see that $\frac{dS_t}{dt} > 0$ for every $t \geq 0$. Therefore, for every $S_0 \geq 3$ the size of the region will be forever growing.

Corollary 1. *Given a region of size S_0 tiles, expanding every d time steps, where $d = O(1)$ w.r.t S_0, then a team of less than $\Omega(\sqrt{S_0})$ robots cannot clean the region, regardless of the algorithm used.*

Theorem 3. *Let F be a region of size S_0 tiles, expanding every d time steps. A team of k robots located at $t = 0$ on the same tile cannot clean F, regardless of the algorithm used, if $d^2k < \Omega(R(F))$, where $R(F)$ is the perimeter of the bounding rectangle of F.*

Proof. For every $v \in F$ let $l(v)$ denote the maximal distance between v and any of the tiles of F, namely:

$$l(v) = \max\{d(v,u) | u \in F\}$$

Let $C(F) = l(v_c)$ such that $v_c \in F$ is the tile with minimal value of $l(v)$.

Let v_s denote the tile the agents are located in at $t = 0$. Let $v_d \in F$ denote some contaminated tile such that $d(v_s, v_d) = l(v_s)$. Regardless of the algorithm used by the agents, until some agent reaches v_d there will pass at least $l(v_s)$ time steps. Let us assume w.l.o.g that v_d is located to the right (or of the same horizontal coordinate) and to the top (or of the same vertical coordinate) of v_s. Then by the time some agent is able to reach v_d there exists an upper-right quarter of a digital sphere of radius $\left\lfloor \frac{l(v_s)}{d} \right\rfloor + 1$, whose center is v_d. The number of tiles in such a quarter of digital sphere equals:

$$\frac{1}{2}\left\lfloor \frac{l(v_s)}{d} \right\rfloor^2 + \frac{3}{2}\left\lfloor \frac{l(v_s)}{d} \right\rfloor + 1 = \Theta\left(\frac{l(v_s)^2}{d^2}\right)$$

It is obvious that the region cannot be cleaned until v_d is cleaned. Let t_d denote the time at which the first agent reaches v_d. It is easy to see that $t_d \geq l(v_s)$. Therefore, at time t_d there are k agents that has to clean a region of at least $\Theta(\frac{l(v_s)^2}{d^2})$ tiles, spreading every d time steps. Using Theorem 2 we know that k agents cannot clean an expanding region of $k = \frac{\sqrt{S_0}}{d}$ tiles. Namely, at time t_d the k agents could not clean the contaminated tiles if:

$$d^2 k < \Omega\left(l(v_s)\right)$$

As $l(v_s) \geq C(F)$ we know that k agents could not clean an expanding contaminated region where: $d^2 k < \Omega\left(C(F)\right)$. It is easy to see that for every region F, if $R(F)$ is the length of the perimeter of the bounding rectangle of F then $C(F) = \Theta(R(F))$.

Lemma 1. *For every connected region of size $S_0 \geq 3$ and perimeter of length c_0:*

$$\frac{1}{2}c_0 < \gamma_2 < c_0$$

Proof. let us assume by contradiction that $(c_0 + 2)^2 \leq (8S_0 + 8)$, implying $c_0 \leq \sqrt{8S_0 + 8} - 2$. However, the minimal circumference of a region of size S_0 is achieved when the region is arranged as an 8-connected digital sphere, in which $c_0 \geq 4\sqrt{S_0} - 4$, contradicting the assumption that $c_0 \leq \sqrt{8S_0 + 8} - 2$ for $S_0 > 5$ and hence, $\gamma_2 \in \mathbb{R}$.

Let us assume by contradiction that $\gamma_2 < \frac{1}{2}c_0$. This in turn implies:

$$c_0 < -\frac{16}{6} + \sqrt{10\frac{2}{3}S_0 - 8\frac{8}{9}} < 3.266\sqrt{S_0} - 2$$

We know that $c_0 \geq 4\sqrt{S_0} - 4$, which contradicts the assumption that $\gamma_2 < \frac{1}{2}c_0$ for every $S_0 \geq 3$. Let us assume by contradiction that $\gamma_2 > c_0$, implying that $c_0 > 4S_0 - 6$.

We know that $c_0 \leq 2S_0 - 2$ *(as* c_0 *is maximized when the tiles are arranged in the form of a straight line), contradicting the assumption that* $\gamma_2 > c_0$ *for every* $S_0 \geq 3$.

Lemma 2. *For every connected region of size* $S_0 \geq 3$ *and perimeter of length* c_0*:*

$$\Omega(1) < \gamma_1 < O(\ln n)$$

Proof. Let us observe γ_1*:* $\gamma_1 \triangleq \psi\left(1 + \frac{c_0+2+\gamma_2}{4}\right) - \psi\left(1 + \frac{c_0+2-\gamma_2}{4}\right)$*. From Lemma 1 we can see that* $1 < \left(1 + \frac{c_0+2-\gamma_2}{4}\right) < \frac{1}{4}c_0$*. Note that* $\psi(1) = -\hat{\gamma}$ *where* $\hat{\gamma}$ *is the* Euler-Mascheroni *constant which equals approximately* 0.57721*. In addition,* $\psi(x)$ *is monotonically increasing for every* $x > 0$*. As* $\psi(x)$ *is upper bounded by* $O(\ln x)$ *for large values of* x*, we see that:*

$$-0.58 < \psi(1 + \frac{c_0 + 2 - \gamma_2}{4}) < O(\ln n) \tag{1}$$

From Lemma 1 we also see that $1 < \left(1 + \frac{c_0+2+\gamma_2}{4}\right) < \frac{1.5}{4}c_0$ *meaning that:*

$$\psi(1 + \frac{c_0 + 2 + \gamma_2}{4}) = \Theta(\ln n) \tag{2}$$

Combining equations 1 and 2 the rest is implied.

Theorem 4. *Result 3 returns a positive real number for the covering time of a region of* S_0 *tiles that expands every* d *time steps, when the number of robots equals* $\Theta(\sqrt{S_0})$ *and* $d = \Omega(\frac{c_0}{\gamma_1})$*, where* γ_1 *defined in Result 3 shifts from* $O(1)$ *to* $O(\ln S_0)$ *as* c_0 *grows from* $O(\sqrt{S_0})$ *to* $O(S_0)$*.*

Proof. In order for Result 3 to yield a real number all the following must hold:

$$d \cdot k \neq 0 \quad , \quad |\partial F| > 1 \quad , \quad A_3 \neq 0 \quad , \quad (c_0 + 2)^2 > 8S_0 - 8$$

$$(A_1A_3 - A_4)^2 \geq 4A_3(A_2 - A_1 - A_1A_4)$$

The first and second requirements hold for every non trivial scenario. The third requirement is implied by the fourth. The fourth assumption is a direct result of Lemma 1.

As for the last requirement, we ask that $A_1^2A_3^2 + A_4^2 \geq 4A_2A_3 - 4A_1A_3 - 2A_1A_3A_4$*, which subsequently means that we must have:*

$$\frac{\gamma_2^2}{d^2k^2}\left(c_0^2 + \gamma_2^2 - c_0\gamma_2\right) + \gamma_1^2 + \frac{\gamma_2^2\gamma^2}{d} - \frac{\gamma\gamma_1\gamma_2}{d} \geq \frac{\gamma_2}{dk} \cdot O\begin{pmatrix} \gamma_2 - c_0\gamma_1 + c_0\frac{\gamma_2\gamma}{d} \\ -\gamma_1 + \gamma_1\gamma_2 - \frac{\gamma_2^2\gamma}{d} \end{pmatrix}$$

Using Lemma 1 and Lemma 2 we should make sure that:

$$\frac{\gamma_2^2}{dk^2}c_0^2 + \gamma_1^2 d + \gamma_2^2\gamma^2 - \gamma\gamma_1\gamma_2 \geq O\left(\frac{c_0\gamma_2\gamma_1}{k} + \frac{c_0\gamma_2^2\gamma}{dk}\right)$$

Using $W(F) = O(\sqrt{S_0})$ *and* $\Omega(\sqrt{S_0}) = |\partial F| = O(S_0)$ *and dividing by* γ_2^2 *(which we know to be larger than zero), we can write the above as follows:*

$$\frac{c_0^2}{dk^2} + \frac{k^2 + d\ln^2 S_0}{c_0^2} + 1 \geq O\left(\frac{\ln S_0}{c_0} + \frac{k\ln S_0}{c_0^2} + \frac{\ln S_0}{k} + \frac{c_0\sqrt{S_0}}{dk^2} + \frac{c_0}{dk} + \frac{1}{d}\right)$$

As $c_0 \geq \sqrt{S_0}$ *then* $\frac{c_0^2}{dk^2} \geq \frac{c_0\sqrt{S_0}}{dk^2}$. *In addition,* $1 \geq \frac{1}{d}$ *and also* $1 \geq \frac{\ln S_0}{c_0}$ *and* $1 \geq \frac{\ln S_0}{k}$. *In order to have also* $1 \geq \frac{c_0}{dk}$ *we must have:* $d \cdot k = \Omega(c_0)$

In addition, we also require that the cleaning time μ *is positive:*

$$A_4 + \sqrt{(A_1A_3 - A_4)^2 - 4A_3(A_2 - A_1 - A_1A_4)} > A_1A_3$$

For this to hold we shall merely require that $A_2 - A_1 - A_1A_4 \leq 0$ *(as* A_3 *is known to be positive). Assigning the values of* A_1, A_2, A_4, *this translates to* $c_0 + c_0^2\frac{\gamma}{d} \leq O(c_0\gamma_1)$.

Dividing by c_0 *we can now write* $c_0 + \frac{c_0\sqrt{S_0}}{k} + k \leq dO(\gamma_1)$. *As* c_0 *is the dominant element, we see that* $d = \Omega\left(\frac{c_0}{\gamma_1}\right)$. *which subsequently implies that:* $\Omega(\sqrt{S_0}) \leq k \leq O(c_0)$. *Therefore, we select the value of* k *such that* $k = \Theta(\sqrt{S_0})$.

Theorem 5. *The time it takes a group of* $k = \Theta(\sqrt{S_0})$ *robots using the* ***SWEEP*** *algorithm to cover a connected region of size* S_0 *tiles, that expands every* $d = \Omega(\frac{c_0}{\gamma_1})$ *time steps (where* γ_1 *is defined in Result 3), is upper bounded by* $O\left(S_0^2 \ln S_0\right)$.

Proof. Recalling Result 1, as we want at least a single contamination spread we assume $\frac{8(|\partial F_0|-1)\cdot(W(F_0)+k)}{k} + 2k \geq d$. *Observing Result 3 we now see that:*

$$t_{SUCCESS} = d \cdot O\left(A_1 + \frac{|A_4|}{A_3} + \sqrt{A_1^2 + \frac{|A_1A_4| + A_1 + A_2}{A_3} + \frac{A_4^2}{A_3^2}}\right) \leq$$

$$d \cdot O(\, c_0 + \gamma_2 + dk\frac{\gamma_1}{\gamma_2} + k\gamma + \sqrt{k}\frac{\sqrt{c_0+\gamma_2}\sqrt{d\gamma_1+\gamma_2\cdot\gamma}}{\sqrt{\gamma_2}} + \sqrt{\frac{kd}{\gamma_2}}\sqrt{c_0+\gamma_2}\,)$$

Using the fact that $\gamma_2 = \Theta(c_0)$ *(Lemma 1) we can rewrite this expression as:*

$$d \cdot O(c_0 + dk\frac{\gamma_1}{c_0} + k\gamma + \sqrt{k}\sqrt{d\gamma_1 + c_0\gamma} + \sqrt{kd}) \tag{3}$$

As $W(F_0) = O(\sqrt{S_0})$, *we can now upper bound* d *as follows:* $d = O(\frac{\sqrt{S_0}\cdot c_0}{k} + c_0 + k)$. *Therefore,* $|\gamma|$ *can now be written as* $|\gamma| = O(\frac{\sqrt{S_0}}{k} + \sqrt{S_0} + \frac{k}{\sqrt{S_0}} + 1)$. *Remembering that* $O(\sqrt{S_0}) \leq c_0 \leq O(S_0)$ *we can rewrite Equation 3 as follows:*

$$d\cdot O\Big(\, c_0 + dk\tfrac{\gamma_1}{c_0} + k\sqrt{S_0} + \tfrac{k^2}{\sqrt{S_0}} + \sqrt{kc_0}\left(\sqrt{\gamma_1} + \sqrt[4]{S_0}\sqrt{\tfrac{\gamma_1}{k}} + \sqrt[4]{S_0} + \tfrac{\sqrt{k}}{\sqrt[4]{S_0}}\right) + \sqrt{kd}\Big)$$

which 2 can simplify to $d\cdot O(c_0 + k\sqrt{S_0}\ln S_0 + \frac{k^2}{\sqrt{S_0}} + \sqrt{c_0\ln S_0\sqrt{S_0}} + \sqrt{c_0k\sqrt{S_0}})$.

Assuming that $k > O(\ln S_0)$ *and as* $c_0 = O(S_0)$ *we can now write:*

$$O\left(\tfrac{S_0^{2.5}}{k} + S_0^2\ln S_0 + S_0^{1.5}k\ln S_0 + k^2\sqrt{S_0}\ln S_0 + \tfrac{k^3}{\sqrt{S_0}} + \tfrac{S_0^{2.25}}{\sqrt{k}} + S_0^{1.75}\sqrt{k} + S_0^{0.75}k^{1.5}\right)$$

5 Conclusions

In this paper we have discussed the covering of a connected region on the grid using collaborate ant-like robotics system. We have shown that for static regions this can be done in $O(\frac{1}{k}\sqrt{n} \cdot c_0 + c_0 + k)$ time, equals $O(\frac{1}{k}n^{1.5} + n)$ time in the worst case. In addition, we have shown that when a region expands in a constant rate, a team of $\Theta(\sqrt{n})$ robots can still be guaranteed to clean or cover it, in $O(n^2 \ln n)$ time.

In addition, we have shown that teams of less than $\Omega(\sqrt{n})$ robots can *never* cover a region that expands every $O(1)$ time steps, regardless of their sensing capabilities, communications and memory resources, or algorithm used. As to regions that expand slower, two impossibility results were shown. First, a region of n tiles that expands every d time steps cannot be covered by a group of k agents if $dk \leq O(\sqrt{n})$. Theorem 4 guarantees a coverage when $dk = \Omega(\frac{n^{1.5}}{\ln n})$, or even for $dk = \Omega(n))$ (when the region's perimeter c_0 equals $O(n)$).

Second, a spreading region cannot be covered when d^2k is smaller than the order of the perimeter of the bounding rectangle of the region (which is $O(n)$ in the worse case and $O(\sqrt{n})$ for "round shapes"). Theorem 4 guarantees a coverage when $d^2k = \Omega(\frac{n^{2.5}}{\ln^2 n})$, or for $d^2k = \Omega(n^{1.5})$ (when $c_0 = O(n)$).

References

1. Altshuler, Y., Bruckstein, A., Wagner, I.: Swarm robotics for a dynamic cleaning problem. In: IEEE Swarm Intelligence Symposium, pp. 209–216 (2005)
2. Altshuler, Y., Wagner, I., Yanovski, V., Bruckstein, A.: Multi-agent cooperative cleaning of expanding domains. The Int. J. of Robotics Res. (to appear, 2010)
3. Altshuler, Y., Yanovsky, V., Bruckstein, A., Wagner, I.: Efficient cooperative search of smart targets using uav swarms. ROBOTICA 26, 551–557 (2008)
4. Borie, R., Tovey, C., Koenig, S.: Algorithms and complexity results for pursuit-evasion problems. In: The Int. Joint Conf. on AI (IJCAI), pp. 59–66 (2009)
5. Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: Robotic exploration as graph construction. IEEE Transactions on Robotics and Automation 7, 859–865 (1991)
6. Isler, V., Kannan, S., Khanna, S.: Randomized pursuit-evasion with local visibility. SIAM Journal of Discrete Mathematics 20, 26–41 (2006)
7. Koenig, S., Liu, Y.: Terrain coverage with ant robots: A simulation study. In: Proc. of the 5th Int. Conf. on Autonomous agents, pp. 600–607 (2001)
8. Koenig, S., Szymanski, B., Liu, Y.: Efficient and inefficient ant coverage methods. Annals of Mathematics and Artificial Intelligence 31, 41–76 (2001)
9. Szymanski, B., Koenig, S.: The complexity of node counting on undirected graphs. Technical Report, CS Department, Rensselaer Polytechnic Institute (1998)
10. Thrun, S.B.: Efficient exploration in reinforcement learning — technical report cmu-cs-92-102. Technical report, Carnegie Mellon University (1992)
11. Wagner, I., Altshuler, Y., Yanovski, V., Bruckstein, A.: Cooperative cleaners: A study in ant robotics. The Int. J. of Robotics Res. 27(1), 127–151 (2008)
12. Wagner, I., Lindenbaum, M., Bruckstein, A.: Efficiently searching a graph by a smell-oriented vertex process. Annals of Math. and AI 24, 211–223 (1998)

The Design of an Active Structural Vibration Reduction System Using a Modified Particle Swarm Optimization

Adam Schmidt

Institute of Control and Information Engineering, Poznan University of Technology, Poznan, Poland
Adam.Schmidt@put.poznan.pl

Abstract. This paper presents the synthesis of an active control system using a modified particle swarm optimization method. The system's controller design is analyzed as a minimalization of the building stories' acceleration. The proposed fitness function is computationally efficient and incorporates the constraints on the system's stability and the maximum output of actuators. In order to handle the constraints the PSO was modified to take into account the particles' distance to the best and the worst solutions. The performance of the obtained controller was tested using historical earthquake records. The performed numerical simulations proved that the designed controller is capable of efficient vibrations reduction.

Keywords: active vibration reduction, particle swarm optimization.

1 Introduction

Nowadays, it is common to design and construct lightweight and cost-efficient buildings. However, these light constructions tend to be susceptible to vibrations caused either by human or by natural sources such as earthquakes. The concept of the active control system[13], is one of the possible solutions to that problem. Numerous methods have been used to design the controller for the active vibration reduction systems. A comprehensive survey of approaches based on both the control theory as well as on the soft computing can be found in [6].

Over the last years, the Particle Swarm Optimization has been successfully used in the controller design. In [8] and [2] the PSO has been used in the optimization of the PI and PID controllers. Wang et al.[11] have applied the PSO to find the control system's poles resulting in a robust control system. In [10] the PSO has been succesfully used to find the optimal feedback gain in the vehicle navigation system controller.

This paper presents the design of the building active control system using a modified PSO. The constrained controller design is formulated as an optimization problem. The proposed fitness function minimizes the building's structure accelerations and incorporates the constraints on the system's stability as well as

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 544–551, 2010.

the requirements concerning maximum forces generated by actuators. In order to adapt the PSO to the constrained optimization the unfeasible solutions are not only attracted to the best solutions but also repeled from the worst solutions. The effectiveness of the presented method is assessed on the model of a six-story building under different earthquakes.

2 Structure Model

The building is modeled as a shear planar frame with actuators installed between some of its stories. It is assumed that the braces supporting actuators are infinitely stiff. The motion equations of the system can be defined as:

$$M\ddot{q}(t) + C\dot{q}(t) + Kq(t) = Eu(t) + M \cdot 1_{N\times 1} a_{gr}(t) \tag{1}$$

where M, C, K and E stand for the mass, damping, stiffness and location matrices. The $q(t)$ is a vector of the stories displacements relative to the ground, $u(t)$ is a vector of the forces generated by the actuators and the $a_{gr}(t)$ is the acceleration of the ground.

The forces generated by the actuators are calculated according to the structure displacements $q(t)$ and velocities $\dot{q}(t)$. It is assumed, that all displacements and velocities are measured, which means that:

$$u(t) = -G_1 q(t) - G_2 \dot{q}(t) \tag{2}$$

where G_1 and G_2 are the gain matrices of the control system feedback loop. The Eqn. 1 can be rewritten as:

$$M\ddot{q}(t) + (C + EG_2)\dot{q}(t) + (K + EG_1)q(t) = M \cdot 1_{N\times 1} a_{gr}(t) \tag{3}$$

The system's model can be rephrased in the form of the state equations:

$$z(t) = \left[q(t)\ \dot{q}(t) \right]^T \tag{4}$$

$$\dot{z}(t) = Az(t) + Ba_{gr}(t) \tag{5}$$

$$A = \begin{bmatrix} 0_{(N\times N)} & 1_{(N\times N)} \\ -M^{-1}(K + EG_1) & -M^{-1}(C + EG_2) \end{bmatrix} \tag{6}$$

$$B = \left[0_{1\times N}\ 1_{1\times N} \right]^T \tag{7}$$

3 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) introduced by Kennedy and Eberhart [5] is a population-based optimization technique inspired by social behaviour of animals e.g. birds flocking or fish schooling. Each particle in the swarm keeps a record of its best fitness achieved so far (along with the associated solution Pb) and the best fitness and corresponding solution achieved in the particle's neighborhood - Lb. At

each iteration i of the PSO the velocities of the particles are changed (accelerated) towards the Pb and the Lb and the particles are moved to new positions:

$$v_j(i) = w \cdot v_j(i-1) + c_1 \cdot r_1 \bullet (Pb_j - p_j(i-1)) + c_2 \cdot r_2 \bullet (Lb_j - p_j(i-1)) \quad (8)$$

$$p_j(i) = p_j(i-1) + v_j(i) \quad (9)$$

where v_j and p_j are the velocity and the position of the j-th particle, w is the inertia weight, c_1 is the individuality constant and c_2 is the sociality constant. In the above equation r_1 and r_2 are uniformly distributed random vectors ($U(0,1)$) and $\bullet$ stands for the element-wise multiplication.

The PSO in its canonical form[5] was designed to solve only unconstrained optimization problems. The lack of intrinsic method for constraints handling led to numerous modifications of the algorithm. Hu and Eberhart[4] proposed initializing the particles only in the feasible positions and using only constraints-abiding solutions as the local and global best positions. However, the necessity to find feasible solutions before the optimization makes this approach unapplicable to problems with complex constraints. Parsopoulos and Vrahatis[7] used a dynamic, problem dependent penalty function. The value of the penalty reflected the degree of constrains violations facilitating particles' movement towards the feasible solutions.

He and Wang presented a co-evolutionary particle swarm optimization[3] (COPSO) which concurrently optimizes the original objective function as well as the penalty factors. The COPSO scheme uses an external swarm of size M, which particles represent penalty factors and M internal swarms solving the original problem using the penalty factors defined by a corresponding external swarm particle.

Sedlaczek and Eberhard noticed that for some problems using the penalty function to handle the constraints would require infinite penalty factors to finds the optimal solution. To avoid it they proposed the augmented lagrangian PSO (ALPSO)[9]. The general Lagrange function is augmented by a penalty factors punishing constraints violations which ensures that the solution of the original problem is a minimum of the lagrangian. The particle swarm is used to minimize the lagrangian with the lagrange multipliers and penalty factors updated every k iterations.

The hybrid multi-swarm PSO proposed by Wang and Cai[12] combines the idea of swarm divisions with the differential evolution (DE). At each iteration the swarm is divided into N sub-swarms in a way that maximizes the diversity of sub-swarms. Then each of them is updated according to the simplified rule of Clerc and Kennedy[1]. After that the personal best of each particle is additionally updated with the DE.

4 Proposed PSO Modification

The canonical PSO with a dynamic, problem dependent penalty function is successful if either at least one of the initial particles' position is feasible or if

any particle drawn to the currently best solutions stops at a feasible position. However, if all particles start in unfeasible positions and their trajectories miss the feasible regions the whole swarm will converge to an unfeasible solution considered to be the best. This phenomena is especially inconvienient in case of problems with complex constraints and relatively small feasible solutions space which makes randomly spotting a feasible solution improbable.

A one possible solution to this problem is to not only attract unfeasible particles to the currently best solutions but also to repel them from the worst known positions. As the penalty function rises proportionally to the degree of the constraints violation repulsing from the worst solution should direct the particles towards the feasible solutions.

Each particle stores the worst fitness achieved so far along with a corresponding position Pw and the worst fitness and position achieved in features neighborhood (Lw). If the current position of a particle is unfeasible the velocity components $v_j^P(i)$ and $v_j^L(i)$ repulsing from the Pw and Lw are calculated:

$$v_j^P(i) = \max(\min(Pw_j - p_j(i-1), R), -R) - R \cdot sign(f_j^P(i)) \tag{10}$$

$$v_j^L(i) = \max(\min(Lw_j - p_j(i-1), R), -R) - R \cdot sign(f_j^L(i)) \tag{11}$$

where R is the maximal range of the repulsion forces along each dimension. The absolute values of the repulsion components are close to R near the worst positions and linearly decrease to 0 as the difference between the current and the worst positions increases. The velocity of the unfeasible particle is calculated as:

$$\begin{aligned} v_j(i) = c_1 \cdot r_1 \bullet (Pb_j - p_j(i-1)) + c_2 \cdot r_2 \bullet (Lb_j - p_j(i-1)) + \\ c_3 \cdot r_3 \bullet v_j^P(i) + c_4 \cdot r_4 \bullet v_j^L(i) + w \cdot v_j(i-1) \end{aligned} \tag{12}$$

where c_3 and c_4 are parameters similar to c_1 and c_2 while r_3 and r_4 are vectors analogous to r_1 and r_2. In the presented experiments the parameters of the modified PSO were empirically selected as $R = 100000$ and $c_3 = c_4 = 2$.

4.1 Fitness Function

The optimization goal was to find the gain matrices G_1 and G_2 that would minimize the accelerations of the building's stories under the earthquake. Additionally, the resulting model had to be stable and the generated forces had to be lower than an assumed value ($F_{max} = 100kN$). Those constraints were incorporated into the fitness function:

$$\text{fit}(p) = \text{fit}_{acceleration}(p) + a \cdot \text{fit}_{stability}(p) + b \cdot \text{fit}_{forces}(p) \tag{13}$$

where a and b are the constraints coefficients.

The accelerations of the structure were analyzed in the frequency domain under the simplifying assumption that the ground acceleration is a sinusoidal signal. The following accelerations transfer function can be defined:

$$H_{acc}(j\omega) = \frac{\ddot{Q}(j\omega)}{A_{gr}(j\omega)} = (M + \frac{1}{j\omega}(C + E \cdot G_2) - \frac{1}{\omega^2}(K + E \cdot G_1))^{-1} M \cdot 1_{N \times 1} \tag{14}$$

The biggest (and thus the most dangerous for the structure) accelerations are generated for the modal frequencies of the resulting system. Therefore, the following fitness function component was defined:

$$\mathrm{fit}_{acceleration}(p) = \max_{i,n} |H^n_{acc}(j\omega_i)| \tag{15}$$

where ω_i is the i-th modal frequency of the closed-loop system and H^n_{acc} is the acceleration transfer function of the n-th story.

The resulting system would be stable if the real parts of all the system's poles were smaller than 0. The fitness function stability component was calculated as:

$$\mathrm{fit}_{stability}(p) = \begin{cases} 1 + \max(\Re(e_i)) - \rho & \text{if } \max(\Re(e_i)) \geq \rho \\ 0 & \text{if } \max(\Re(e_i)) < \rho \end{cases} \tag{16}$$

where e_i is the i-th eigenvalue of the state matrix A and ρ is the maximal allowed real part of the system's poles.

It was assumed that the actuators should not saturate until the ground acceleration amplitude reached a certain value ($A_{max} = 0.5\frac{m}{s^2}$) at any of the system's modal frequencies. The following transfer function was defined:

$$H_{force}(j\omega) = \frac{U(j\omega)}{A_{gr}(j\omega)} = (-G_1 - jG_2\omega)H_{disp}(j\omega) \tag{17}$$

The force component of the fitness function was calculated according to:

$$\mathrm{fit}_{force}(p) = \begin{cases} \max\left(\frac{|H_{force}(j\omega_i)|A_{max}}{F_{max}}\right) & \text{if } \max\left(\frac{|H_{force}(j\omega_i)|A_{max}}{F_{max}}\right) > 1 \\ 0 & \text{if } \max\left(\frac{|H_{force}(j\omega_i)|A_{max}}{F_{max}}\right) \leq 1 \end{cases} \tag{18}$$

The a parameter was set to 1000000 and the b was set to 1000. This ensured that any solution resulting in an unstable system would have higher fitness function value than any of the stable ones and that solutions violating the maximum force limits would have worse fitness than those conforming to both constraints.

The proposed fitness function can be calculated without the time consuming simulations which is an important advantage in any iterative optimization algorithm.

5 Results

The proposed modified PSO was compared to the canonical PSO with a penalty function. In both cases the optimization process was executed 100 times. The convergence of the PSO was assumed if the best fitness value in the population had not changed over $i_{conv} = 100$ iterations and the best solution conformed to both constraints.

The modified PSO algorithm converged to a feasible solution 96 times, whereas only 5 runs of the canonical PSO were successful. The proposed algorithm needed averagely 675 iterations to converge (median = 543). The the best result

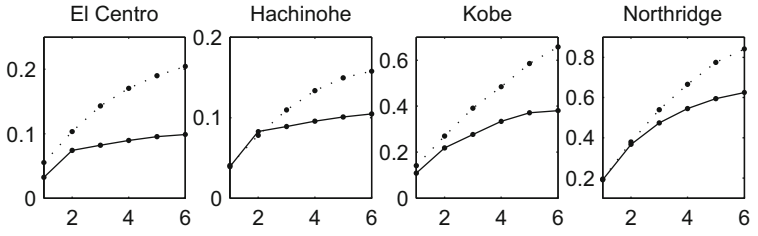


Fig. 1. The maximum displacement of building stories (solid line - controlled, dotted - uncontrolled

obtained by the modified PSO was equal to 1.1287, average fitness function of the converged solutions was equal to 1.5449 with the standard deviation of 0.5184 .

The performance of the obtained controller was tested in numerical simulations. Four different earthquake records were used: El Centro, Hachinohe, Kobe, Northridge. The peak ground accelerations of these earthquakes were: $0.3188g$, $0.2294g$, $0.8337g$, $0.8428g$ respectively.

The maximum displacement of the building stories is shown in the Figure 1. The controler was able to reduce the maximal displacement of the upper (3 to 6) floors of the building significantly. It is especially important as those displacements are the biggest and therefore the most dangerous to the structure.

The examples of the top story displacement of both controlled and uncontrolled building as well as the controller outputs (before saturation) are shown in the Figure 2. In both cases the amplitude and the frequency of the oscillations are greatly reduced. The vibrations in the controlled building are also dampened to 0 faster.

It is worth noting that the controller's outputs did not exceed the F_{max} under the El Centro earthquake which peak ground acceleration was lower than the assumed A_{max} (Eqn. 18). In case of the Kobe earthquake the peak ground acceleration was higher than the A_{max} which caused the actuators to saturate.

Additionally, the normed RMS of structure displacement and the normed RMS fo the total structure acceleration were calculated for all the considered earthquakes (Table 1):

$$D_{RMS} = \frac{\max_i \sqrt{\frac{1}{T}\int_0^T q_i^c(t)^2 \mathrm{dt}}}{\max_i \sqrt{\frac{1}{T}\int_0^T q_i^{uc}(t)^2 \mathrm{dt}}} \tag{19}$$

$$A_{RMS} = \frac{\max_i \sqrt{\frac{1}{T}\int_0^T \left(\ddot{q}_i^c(t) + a_{gr}(t)\right)^2 \mathrm{dt}}}{\max_i \sqrt{\frac{1}{T}\int_0^T \left(\ddot{q}_i^{uc}(t) + a_{gr}(t)\right)^2 \mathrm{dt}}} \tag{20}$$

where i is the story number, T is the duration time of the earthquake, $q_i^c(t)$ and $\ddot{q}_i^c(t)$ are the displacement and acceleration of the i-th story of the controlled building and $q_i^{uc}(t)$ and $\ddot{q}_i^{uc}(t)$ stand for the displacement and acceleration of

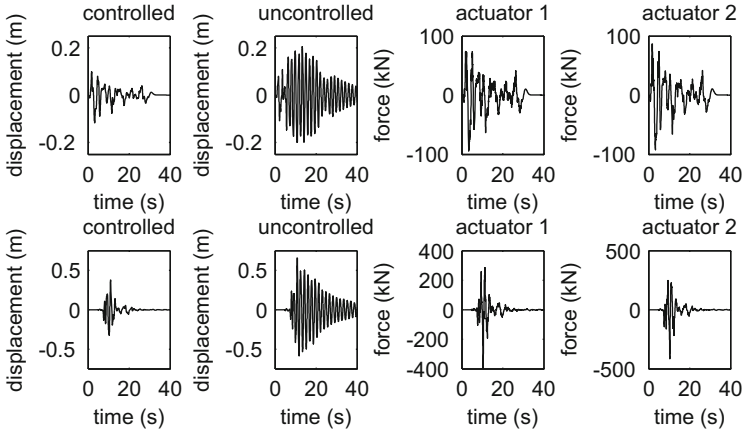


Fig. 2. The displacement of the top floor and the controller output under the El Centro (top) and Kobe (bottom) earthquakes

Table 1. The normalized RMS of building displacement

	El Centro	Hachinohe	Kobe	Northridge
D_{RMS}	0.3188	0.4413	0.3267	0.5679
A_{RMS}	0.3009	0.2457	0.3664	0.4638

the i-th story of the uncontrolled building. It is clearly visible that the vibration control system was able to reduce the RMS of both stories' displacements and accelerations significantly.

6 Conclusions

The presented study shows that the swarm intelligence can be succesfully used in the design of the active vibration reduction systems. The proposed modifications of the PSO algorithm facilitated finding feasible solutions under complex constraints.

A novel, computationally efficient fitness function minimizing the building's stories accelerations and incorporating the control system constraints was defined. The application of the Fourier transform allowed evaluating the solution without the need for costly simulations.

The obtained controller was tested under different historical earthquake loads. It achieved excellent results in the reduction of the maximum stories displacements as well as the normalized RMS of the stories' displacements and total accelerations. Moreover, the obtained solutions conforms to the constraints despite the assumed modelling simplifications.

The future research will focus on modifying the fitness function to take into account the accelerations, velocities and displacements of the building stories

simultaneously. Additionaly, the number and positions of sensors and actuators will be incorporated into the optimization scheme. Moreover, the proposed modification of the PSO algorithm will be applied to different engineering problems and systematical study of the method's parameters will be conducted.

References

1. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
2. Gaing, Z.L.: A particle swarm optimization approach for optimum design of PID controller in AVR system. IEEE Transaction on Energy Conversion 19(2), 384–391 (2004)
3. He, Q., Wang, L.: An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Engineering Applications of Artificial Intelligence 20, 89–99 (2007)
4. Hu, X., Eberhart, R. C.: Solving constrained nonlinear optimization problems with particle swarm optimization. In: Proceedings of the sixth world multiconference on systemics, cybernetics and informatics 2002 (2002)
5. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
6. Lewandowski, R.: A survey of modern methods of vibration reduction of building structures. Vibrations in Physical Systems 22, 37–48 (2006)
7. Parsopoulos, K.E., Vrahatis, M.: Particle swarm method for constrained optimization problems. In: Proceedings of the Euro-international symposium on computational intelligence (2002)
8. Qiao, W., Venayagamoorthy, G., Harley, R.: Design of optimal PI controllers for doubly fed induction generators driven by wind turbines using particle swarm optimization. In: Proceedings of 2006 International Joint Conference on Neural Networks, pp. 1982–1987 (2006)
9. Sedlaczek, K., Eberhard, P.: Using augmented lagrangian particle swarm optimization for constrained problems in engineering. Struct. Multidisc. Optim. 32, 277–286 (2006)
10. Sun, T.Y., Huang, C.S., Tsai, S.J.: Particle swarm optimizer based controller design for vehicle navigation system. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 909–914 (2008)
11. Wang, J., Brackett, B., Harley, R.: Particle swarm-assisted state feedback control: From pole selection to state estimation. In: 2009 American Control Conference, pp. 1493–1498 (2009)
12. Wang, Y., Cai, Z.: A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems. Front. Comp. Sci. China 3(1), 38–52 (2009)
13. Yao, J.T.P.: Concept of structural control. ASCE J. Struct. Div. 98, 1567–1575 (1972)

Ant Colony Extended: Search in Solution Spaces with a Countably Infinite Number of Solutions

Jose B. Escario, Juan F. Jimenez, and Jose M. Giron-Sierra

Arquitectura de Computadores y Automatica Dept.
Universidad Complutense, Madrid, Spain
jbescario@filos.ucm.es, {juan.jimenez,gironsi}@fis.ucm.es

Ant Colony Extended (ACE) is a new framework that allows to apply the ant colony paradigm [1] to solve combinatorial optimization problems, in which the values of each variable are taken from a finite set, and the set of possible solutions is *countably infinite*.

Previously, we applied ACE to autonomous ship manoeuvre planning [2], where the objective was to minimize the time of a manoeuvre. In this problem, as in the TSP and others, the value of the cost function increases monotonically as new elements are added to the solution sequence. We want to check the algorithm for problems that do not exhibit this feature. For this purpose we select a set of multi-modal functions to minimize with ACE: Griewank's function (F2), Shekel's foxholes (F3), Michalewicz' function (F4) and Langerman's function (F5). All functions are taken from [4].

These functions have many local minima, where algorithms may get stuck. We select the Simple Genetic Algorithm (SGA), and Differential Evolution (DE) to perform a comparison of local minima avoidance.

1 Harvester Ants: How to Set a Search Zone

We copy the strategy described by D. Gordon [3] on how harvesting ants set their foraging areas. These areas are the places where ants gather food and they change from day to day.

In order to reproduce this strategy we employ two kinds of ants: a)*Foragers:* ants that follow the pheromone, b)*Patrollers:* ants in charge of the exploration process.

Therefore, patrollers provide the search zones, while foragers select from the available zones the most promising ones.

2 Results and Discussion

A number can be seen as an expression that is generated according to some grammar. ACE can handle an expression optimization –a combinatorial optimization– building the numbers digit to digit. For example the generation of number 100: $1 \rightarrow 10 \rightarrow 100$.

To find the minimum of each function, we perform a batch of 1000 independent executions. We measure success rates: number of optimal values found over the

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 552–553, 2010.

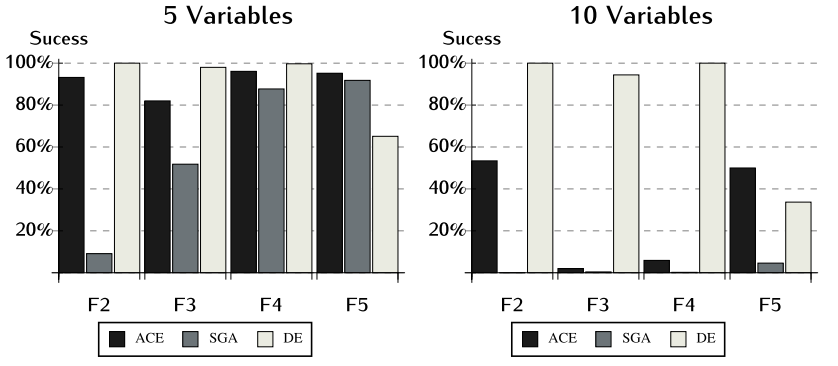


Fig. 1. Comparison results for the set of functions in the case of 5 and 10 variables

number of executions, expressed as percentages. Figure 1 shows the results for the cases of 5 and 10 variables, setting the limit for the number of function evaluations to 400000 and 3000000 respectively.

The comparison with SGA suggests that both algorithms can get results while the difficulty is moderate. When the difficulty increases it seems that ACE may handle it better depending on the scenario. In comparison with DE, although ACE has a worse performance, it has a better behaviour for F5. We should note that DE is an algorithm designed for numerical continuous optimization.

The results prove that ACE is able to solve problems where the value of the cost function does not increase monotonically as new elements are added to the solution sequence.

References

1. Dorigo, M., Stützle, T.: Ant colony optimization. The MIT Press, Cambridge (2004)
2. Escario, J.B., Jimenez, J.F., Giron-Sierra, J.M.: Autonomous ship manoeuvring planning based on the ant colony optimization algorithm. In: Proceedings of 8th Conference on Manoeuvring and Control of Marine Craft, MCMC 2009 (2009)
3. Gordon, D.: Ants at work: how an insect society is organized. Free Press, New York (1999)
4. Seront, G., Gambardella, L.: Results of the first international contest on evolutionary optimisation. In: Proceedings of 1st ICEO IEEE International Conference on Evolutionary Computation, pp. 611–615 (1996)

Automatic Parameter Configuration of Particle Swarm Optimization by Classification of Function Features

Tjorben Bogon, Georgios Poursanidis, Andreas D. Lattner, and Ingo J. Timm

Information Systems and Simulation Institute of Computer Science and Mathematics,
Goethe University Frankfurt, Germany
{tbogon,lattner,timm}@cs.uni-frankfurt.de

Metaheuristics in stochastic local search are used in numerical optimization problems in high-dimensional spaces. A characteristic of these metaheuristics is the configuration of the parameters. These parameters are essential for the optimization behavior but depend on the objective function. In this paper we introduce a new approach to automatic parameter configuration of Particle Swarm Optimization (PSO) by classifying features of the objective function. This classification utilizes a decision tree that is trained by 32 different function features. These features result from the characteristics of the underlying function landscape and of the PSO behavior. An efficient set of parameters influences the optimization in speed and performance. In literature standard configurations are introduced for different types of metaheuristics which perform a not optimal but an adequate optimization behavior for most objective functions. PSO is an example for the parameter configuration problem [2]. The swarm behavior depends mainly on the chosen parameter and leads to solutions of different quality, i.e. bad parameter sets can lead to a disadvantageous balance between exploitation and exploration. One problem by choosing the right parameter without knowledge about the objective function is to describe the characteristics of the function which are comparable to another function.

Similar to Leyton-Brown et al. [1] we create features to describe a function. With these features we train a classification tree representing characteristics of the fitness function landscape. These features are computed and combined with the best found parameter set on the function to a training set (Figure 1). With a trained classifier at hand we compute the features of the objective function prior to the start of the optimization process. The model, which underlies the decision tree, classifies the function and returns the specific parameter set that is expected to perform better in the optimization process than the optimization with standard parameter. The model determines the parameter set without a priori knowledge about the functions if we have a well-trained classifier. In our first experiments, which we understand as a proof of concept, we have selected only a few functions which do not represent any specific types of function. We want to show that our technique is able to identify functions based on the provided features and thereby predict the specific parameter configuration. In order to learn the classifier which suggests the parameter configuration, different function features are computed.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 554–555, 2010.

We split our function features into three groups. Each group implies a distinct way of collecting information about the fitness topology of the objective function. The group *Random Probing* describes features which are calculated based on a random selection of fitness values and provides a general overview of the fitness topology. Distance-based features are calculated for the group *Incremental Probing*. They reflect the distribution of surrounding fitness values of some pivot elements. The third group of features utilizes the dynamics of PSO to create features by using the changes of the global best fitness within a small PSO instance. The features are scale independent, i.e., that scaling the objective function by constants will not affect the feature values. These features are the basis of our training instances for a C4.5 decision tree which is used as classifier. Experimental trials verify that our decision tree classifies functions correctly into classes that are associated to parameter sets for which the PSO performs a better optimization compared to the standard configuration. In our implementation we use *WEKA's J4.8* implementation. As training set we compute 300 independent instances for each function. One instance consists of 32 function features. The decision tree is created based upon the training data and evaluated by stratified 10-fold cross-validation. Upon the six distinct classes we evaluate the model repeatedly through cross validation. The validation indicates that the model has a mean accuracy of 84.32 percent with a standard deviation of 0.29. Our experiments demonstrated that we are able to classify different functions on basis of a few fitness evaluations and get a parameter set which leads the PSO to a significant better optimization performance.

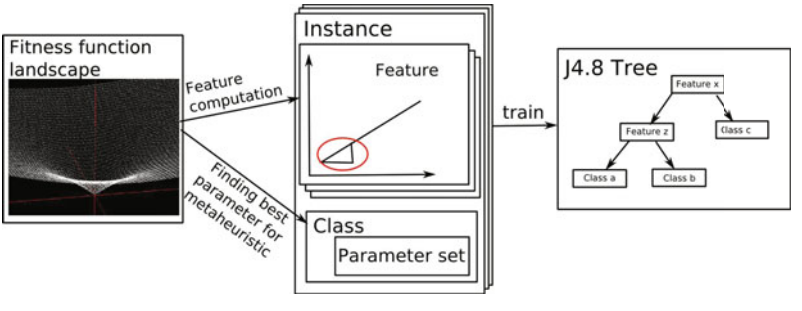


Fig. 1. Feature computation and J4.8 decision tree learning

References

1. Leyton-Brown, K., Nudelman, E., Shoham, Y.: Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 91–100. Springer, Heidelberg (2002)
2. Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 591–600. Springer, Heidelberg (1998)

Constructing Low-Cost Swarm Robots That March in Column Formation

Asuki Kouno[1], Shigeru Takano[2], and Einoshin Suzuki[1,2]

[1] Grad. School of Systems Life Sciences, Kyushu University, Fukuoka, Japan
asu00798@gmail.com, suzuki@inf.kyushu-u.ac.jp
[2] Dept. Informatics, ISEE, Kyushu University, Fukuoka, Japan
takano@inf.kyushu-u.ac.jp

Formation control is an important research topic in swarm robotics, as it enables each swarm robot to concentrate on its task by relying on others [1,2]. This paper is a brief summary of our endeavor for constructing low-cost swarm robots that march in column formation [4]. $n = 5$ swarm robots move in file, except the leading one each follows another one and all of them avoid perimeters of an indoor arena. We describe detailed settings of the control program for further studies. Our work focuses on real robots and is thus considered to be complementary to theoretical and simulation studies, cf. [3].

Each swarm robot [4] is sized approximately 19.0 cm (length) × 18.0 cm (width) × 15.5 cm (height) and costs about 650 US dollars. It is equipped with 5 IR proximity sensors, 1 image sensor, 1 MPU, 2 DC motors, 2 caterpillars, and 1 LED unit. The IR proximity sensor detects obstacles located from 5 cm to 50 cm with precision 1-5 cm, though we assume it is 10 cm for safety. The image sensor provides a color image of 20 pixels × 15 pixels due to the 8KB-size RAM of the MPU, and is the reason of our use of the LED unit to indicate the position of a swarm robot to its follower. The driving unit allows the robot to move forward/backward at velocity 10cm/s - 16cm/s and make a left/right pivot turn at 51°/s - 90°/s. The variations are mainly due to the low precision in its fabrication. Neither communication device, e.g., WIFI, Zigbee, nor positioning device, e.g., GPS, RFID, is used.

The control program [4], which makes no assumption on its environment, enables a swarm robot to move basically forward by avoiding the perimeters if it is the leading one or to follow the preceding swarm robot. It is a function of $P_{\mathrm{LED}} \times R_{\mathrm{IR}} \times S_{\mathrm{IR}} \rightarrow M$, where $P_{\mathrm{LED}}, R_{\mathrm{IR}}, S_{\mathrm{IR}}, M$ represent the position of the preceding swarm robot, the maximum reading value of the IR proximity sensor, the IR proximity sensor that returned the maximum reading value, and the kind of motion, respectively. The preceding robot is indicated by the position of its LED unit along the horizontal axis of the image. We use 2 parameters $J_{\mathrm{out}}, J_{\mathrm{in}}$ $(0 < J_{\mathrm{out}} < J_{\mathrm{in}} < 19)$ to divide the axis into 5 bins (the 4 segmenting points are $J_{\mathrm{out}}, J_{\mathrm{in}}, 19 - J_{\mathrm{in}}, 19 - J_{\mathrm{out}}$) so the domain of P_{LED} is the 5 bins and a null value. The nearer an obstacle is, the larger the reading value of the IR proximity sensor is. We use 3 parameters $I_{\mathrm{avoid}}, I_{\mathrm{near}}, I_{\mathrm{far}}$ $(5 \leq I_{\mathrm{avoid}} \leq I_{\mathrm{near}} \leq I_{\mathrm{far}} \leq 50)$ to divide the distance into at most 4 bins so the domain of R_{IR} is the 4 bins. The domains of S_{IR} and M are the 5 IR proximity sensors and {move forward, move backward, stop, left turn, right turn}, respectively.

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 556–557, 2010.

Table 1. Performances $F_{\rm rate}, K$ of controllers with different values of parameters

	P_{01}	P_{02}	P_{03}	P_{04}	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}	P_{41}	P_{42}
$J_{\rm out}$	3	3	3	3	2	2	1	1	4	4	5	5
$J_{\rm in}$	6	6	6	6	5	5	3	3	7	7	8	8
$I_{\rm far}$	25	35	45	45	25	35	25	35	25	35	25	35
$I_{\rm near}$	15	25	35	25	15	25	15	25	15	25	15	25
$I_{\rm avoid}$	5	15	25	5	5	15	5	15	5	15	5	15
$F_{\rm rate}$	0.93	0.93	0.73	0.16	0.96	0.14	0.86	0.62	0.15	0.16	0.25	0.66
K	1.93	1.66	0.08	0.25	2.30	0.31	2.66	1.95	0.18	0.21	0.05	0.26

Let $F(i,t)$ be the number of robots in the column formation at time t if the robot i is the leading one or 0 otherwise and let $V_i(t)$ be its velocity. The performance of the swarm is evaluated in terms of the quality $F_{\rm rate} \equiv \sum_{t=1}^{T}\sum_{i=0}^{n-1} F(i,t)/(Tn-T)$ of the shape of a formation and its weighted velocity $K \equiv \sum_{t=1}^{T}\sum_{i=0}^{n-1} V_i(t)F(i,t)/\sum_{t=1}^{T}\sum_{i=0}^{n-1} F(i,t)$ over time $t = 1, 2, \ldots, T$. The swarm robots moves in an indoor arena of size 240 cm $\times$ 170cm, surrounded by walls and thermocol blocks and lighted by fluorescent lamps [4,5]. An overhead camera and an external PC are used to generate the log file which contains the sequence of their positions automatically [4,5]. The five robots are covered with sheets of papers of different colors for identification. Noise due to luminance diversity and the environment affects the sensors considerably and is hard to model, making our task challenging and valuable. Table 1, which shows the results of experiments ($290 \leq T \leq 310$), indicates that control programs $P_{01}, P_{02}, P_{11}, P_{21}$ have superior performances. We quantitatively understand that the swarm robots should neither located distant nor turn too frequently.

Acknowledgments. A part of this research was supported by Strategic International Cooperative Program funded by Japan Science and Technology Agency.

References

1. Balch, T., Arkin, R.C.: Behavior-based Formation Control for Multi-robot Teams. IEEE Trans. Robotics and Automation 14(6), 926–939 (1998)
2. Gazi, V., Fidan, B.: Coordination and Control of Multi-agent Dynamic Systems: Models and Approaches. In: Şahin, E., Spears, W.M., Winfield, A.F.T. (eds.) SAB 2006. LNCS, vol. 4433, pp. 71–102. Springer, Heidelberg (2007)
3. Jakobi, N., Husbands, P., Harvey, I.: Noise and The Reality Gap: The Use of Simulation in Evolutionary Robotics. In: Morán, F., Merelo, J.J., Moreno, A., Chacon, P. (eds.) ECAL 1995. LNCS, vol. 929, pp. 704–720. Springer, Heidelberg (1995)
4. Kouno, A.: Realization of Pursuing Actions by Non-communicative Swarm Robots under Unknown Environment. Bachelor dissertation, Informatics Course, Dept. Physics, Fac. Sci., Kyushu Univ., Fukuoka, Japan (March 2010) (in Japanese)
5. Suzuki, E., Hirai, H., Takano, S.: Toward a Novel Design of Swarm Robots Based on the Dynamic Bayesian Network. In: Advances in Data Management. Springer Studies in Computational Intelligence, vol. 223, pp. 299–310. Springer, Heidelberg (2009)

Coordinating Heterogeneous Swarms through Minimal Communication among Homogeneous Sub-swarms

Carlo Pinciroli[1], Rehan O'Grady[1], Anders L. Christensen[2], and Marco Dorigo[1]

[1] IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{cpinciro,rogrady,mdorigo}@ulb.ac.be
[2] Instituto de Telecomunicações, Lisbon, Portugal
anders.christensen@iscte.pt

In swarm robotics, the agents are often assumed to be identical. In this abstract, we argue that the cooperation between swarms of different kinds of robots can enhance the capabilities of the robotic system—heterogeneous swarms marry the robustness and parallelism of homogeneous swarms with efficient task specialisation. A key issue in heterogeneous swarm systems is the potential complexity of facilitating cooperation between the different robot types.

We propose an approach to heterogeneous system design that minimises the complexity of heterogeneous interaction whilst preserving the benefits of specialisation. To mitigate this complexity, we restrict interactions between homogeneous sub-swarms to (very) simple forms of communication. This restriction allows the system to be completely modular. At the top level, modules are global-level behaviours executed by the heterogeneous robotic swarm. Each global-level behaviour is obtained by decomposing the heterogeneous swarm into its homogeneous constituents, which, in turn, execute specific behaviours. In this way, coordinated heterogeneous behaviours can be obtained through minimal inter-sub-swarm communication, even when imprecise information is exchanged.

To demonstrate our approach, we consider a case study of a heterogeneous robotic task in which a swarm of aerial robots (*eye-bots*) recruits groups from a swarm of wheeled robots (*foot-bots*) and sends the groups to locations where tasks need to be executed.

The experimental arena is depicted in Fig. 1. It is formed by four rooms where the eye-bots discover tasks and coordinate their execution, performed by the foot-bots. The foot-bots are initially deployed in the *recruitment area*, that in our setup happens to be located in the centre of the environment. In the recruitment area, eye-bots coordinate the formation of groups of foot-bots. As explained in more detail in [1], the formation of the groups happens in parallel and is completely deadlock-free even when the ground-based robots are fewer than the total needed for all the tasks.

The recruitment area and the task rooms are connected by corridors. In the corridors, further sets of eye-bots serve two purposes: (i) they work as message relayers between the task rooms and the recruitment area and (ii) they guide the groups of foot-bots to their destination using the assisted flocking algorithm described in [2].

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 558–559, 2010.

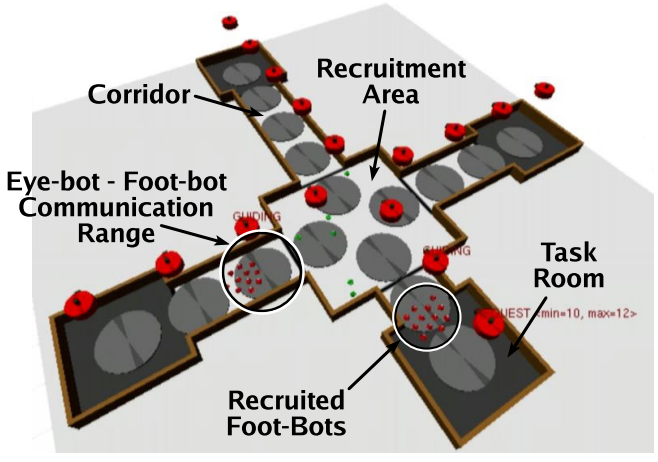


Fig. 1. The experimental arena. The large red blobs are the aerial robots (eye-bots). The smaller blobs (some red, some green) are the ground-based robots (foot-bots).

Results show that the system provides flexibility and allows for the recruitment and delivery of groups of robots in a highly dynamic application scenario. Video footage of the experimental runs is available at `http://iridia.ulb.ac.be/supp/IridiaSupp2010-006/`.

Future work will be devoted to testing the described system on the real robots and to applying our minimal inter-swarm communication paradigm to other complex open problems, such as collective structure building.

Acknowledgements. This research was carried out in the framework of Swarmanoid, a project funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission under grant IST-022888. Marco Dorigo acknowledges support from the Belgian F.R.S.-FNRS, of which he is a Research Director. This study was supported by FCT grant PTDC/EEA-CRO/104658/2008.

References

1. Pinciroli, C., O'Grady, R., Christensen, A.L., Dorigo, M.: Self-organised recruitment in a heterogeneous swarm. In: Prassel, E., et al. (eds.) The 14th International Conference on Advanced Robotics (ICAR 2009). Proceedings on CD-ROM, paper ID 176, p. 8 (2009)
2. Pinciroli, C., O'Grady, R., Christensen, A.L., Dorigo, M.: Wisdom of swarms: A case study in robot navigation. Tech. Rep. TR/IRIDIA/2010-008, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2010)

Effect of Particle Initialization on the Performance of Particle Swarm Niching Algorithms

Isabella Schoeman and Andries P. Engelbrecht

Department of Computer Science, University of Pretoria, South Africa
engel@cs.up.ac.za

The vector-based PSO (VBPSO) [3,4,5] was developed to locate multiple solutions to multi-modal optimization problems. Three versions of the VBPSO were published, and shown to be very efficient in locating mutliple optima. This is despite the fact that the VBPSO algorithms initialize particles using standard pseudo random number generators. The main objective of this article is to show that the perfomance of the VBPSO algorithms can be improved by initializing particles using Sobol sequences [1,2].

Main objectives of niching algorithms are to identify candidate solutions and to demarcate the portion of the search space – called a niche – where an optimal solution may be found. The VBPSO algorithms use the vector dot product, $\delta_i(t) = \mathbf{v}_{pi}(t) \bullet \mathbf{v}_{gi}(t)$ between the cognitive component $\mathbf{v}_{pi}(t) = \mathbf{y}_i(t) - \mathbf{x}_i(t)$ and the social component $\mathbf{v}_{gi}(t) = \hat{\mathbf{y}}(t) - \mathbf{x}_i(t)$ to determine niche boundaries; $\mathbf{x}_i(t), \mathbf{y}_i(t)$ and $\hat{\mathbf{y}}(t)$ respectively refer to the position vector, personal best position, and global best position of particle i at time step t.

If the dot product, $\delta_i(t)$, is positive, then the two vectors, $\mathbf{v}_{pi}(t)$ and $\mathbf{v}_{gi}(t)$, point roughly in the same direction. This means that the personal best and global best positions of a particle move in the same direction. All particles within a certain niche radius and with positive dot products therefor form one niche. A niche boundary is detected as soon as a negative dot product has been found. A candidate niche is therefor located around the current global best position, $\hat{\mathbf{y}}(t)$. For each particle a dot product, $\delta_i(t)$ amd a radius, $\rho_i(t)$ as the distance between the particle's position, $\mathbf{x}_i(t)$, and the global best position, $\hat{\mathbf{y}}(t)$ are computed. At each iteration a dynamic niche radius, $\tau(t)$, is computed as the distance between $\hat{\mathbf{y}}(t)$ and the nearest particle with $\delta_i(t) < 0$ (indicating a niche boundary). All particles with $\rho_i(t) < \tau(t)$ and $\delta_i(t) > 0$ are grouped together with $\hat{\mathbf{y}}(t)$ to form a sub-swarm around the candidate solution. These particles are removed from the main swarm. New sub-swarms are created from the particles that remain in the main swarm and particles within sub-swarms refine their respective candidate solutions. In the case that a sub-swarm does not have at least three particles, extra particles are created around the best solution, within the niche radius.

Another goal of niching algorithms is the process of maintaining niches. The sequential VBPSO (sVBPSO) [3] refines niches in sub-swarms using the standard velocity and position update equations. This may result in duplicate solutions with multiple sub-swarm converging on the same solution. The parallel VBPSO (pVBPSO) [4] merges sub-swarms that converge on the same solution, thereby

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 560–561, 2010.

eliminating duplicate solutions. The enhanced VBPSO (eVBPSO) [5] inhibits the tendency of particles to move outside the bounds of a niche.

The performance of the three VBPSO algorithms were evaluated using a pseudo-random number generator (PRNG) as well as Sobol sequences on the following functions: Himmelblau, Rastrigin, Griewank, Ackley, Ursem F3 and Six Hump Camel, all in two dimensions. PRNG outperformed Sobol sequences only in the accuracy of the solutions obtained for the six hump camel function using pVBPSO. For all other functions, Sobol sequences either performed significantly better than PRNG or the two initialization schemes had similar performance for all performance criteria. The criteria included the average accuracy of all solutions, the number of solutions obtained, and the success rate.

For all the algorithms, Sobol sequences outperformed PRNG for most of the functions with respect to the success rate and the number of solutions found. Sobol sequences were better using sVBPSO for the Himmelblau function (the only function used to evaluate sVBPSO), and with reference to pVBPSO and eVBPSO for all six functions. Therefor, Sobol sequences are more successfull at locating more optima than using a PRNG. With reference to the accuracy of the found solutions, it was predominatly the case that there is no significant difference in performance. However, Sobol sequence did provide more accurate solutions for two functions when pVBPSO was used and for one function when eVBPSO was used. With respect to the number of function evaluations to reach a 98% success rate, there was no significant difference in performance.

In summary, it benefits the performance of the VBPSO algorithms to initialize particles using Sobol sequences instead of the system supplied RNG, specifically with reference to the number of optima found.

References

1. Bratley, P., Fox, B.: Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator. ACM Trans. on Math. Softw. 14, 88–100 (1988)
2. Joe, S., Kuo, F.: Remark on Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator. ACM Trans. on Math. Softw. 29, 49–57 (2003)
3. Schoeman, I., Engelbrecht, A.: Using Vector Operations to Identify Niches for Particle Swarm Optimization. In: Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems, pp. 361–366 (2004)
4. Schoeman, I., Engelbrecht, A.: A Parallel Vector-Based Particle Swarm Optimiser. In: Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, pp. 268–271 (2005)
5. Schoeman, I., Engelbrecht, A.: Containing Particles Inside Niches when Optimising using Multimodal Functions. In: Proceedings of SAICSIT, pp. 78–85 (2005)

Energy Efficient Swarm Deployment for Search in Unknown Environments

Timothy Stirling and Dario Floreano

Laboratory of Intelligent Systems (LIS), Ecole Polytechnique Fédéral de Lausanne (EPFL), Lausanne, Switzerland
tim.stirling@epfl.ch

This paper introduces three strategies to deploy a swarm of robots in unknown environments for a search task, aiming to reduce the total swarm energy cost with rapid operation for applications such as disaster mitigation. We are motivated by current research on flying robot swarms [10].

A complex problem in swarm robotics is controlling deployment in unknown environments. If robots deploy to unnecessary locations, energy is wasted. Conversely, if an area receives insufficient robots the task may be unachievable or performance reduced. In work by Rybski *et al.* [9], increasing the number of deployed robots increased performance by decreasing amounts until a peak was reached, after which the returns diminished. Moreover, Rosenfeld *et al.* [8] noted that after a peak in performance was reached, additional robots often decreased performance due to spatial constraints and interference. Previous work in efficiently coordinating multi-robot search relied upon building environment maps [1,3,5], complex or CPU intensive coordination [11,3], centralised processing [1], globalised coordination and negotiation [5,11] and/or high bandwidth communication [1,11]. Our aerial robots, however, need simple distributed swarm deployment strategies that have minimal communication and processing requirements and do not require environment maps. They have severely limited flight autonomy (e.g. 10-15 minutes [7]). Additionally, rapid deployment is desirable but often there is a trade-off between energy efficiency and time [4]. Furthermore, time and energy were previously either examined in isolation, or only with multiobjective functions that mask details in the individual metrics [2].

We utilise our earlier work on aerial swarm search of unknown indoor environments [10]. Flying robots progressively build a dynamic sensor network by perching on the ceiling to save energy [7]. Environments are systematically searched with depth-first search (see the online video[1]). We compared three swarm deployment strategies: 1) The simplest strategy requires no communication or additional sensing and computation is minimal. Robots deploy one at a time with a fixed time interval between consecutive launches. 2) The second strategy requires no additional sensing but minimal bandwidth communication, either global, or local communication propagated through the robot network. Flying robots emit a signal which is used to ensure only a single robot flies at a time, preventing wasted locomotion [3]. 3) The third strategy requires no communication but a sensor that can perceive the range to nearby robots is used

[1] http://lis.epfl.ch/~stirling/videos/Swarm_Search.avi

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 562–563, 2010.

to calculate the density of flying robots [6]. This density indicates robot congestion and is used with threshold-based task allocation to control the number of concurrently flying robots to reduce interference and wasted locomotion.

We have developed a complete energy model, validated on flying robots [7], which has been used in a 3-D dynamics simulator. Preliminary results indicate that all strategies reduce swarm energy consumption. Launching robots at specified periods permitted the trade-off between energy efficiency and rapid search. Ensuring that only a single robot flies reduces energy costs but significantly slows the search. Controlling the density of robots minimised energy consumption and also achieved the fastest search time. This was due to the ability to dynamically adapt to local environmental and robot congestion conditions. Future work involves extensive simulation analysis and hardware testing.

Acknowledgements. This work is part of the "Swarmanoid Project", a Future Emerging Technologies (FET IST-022888) project funded by the EC.

References

1. Burgard, W., Moors, M., Stachniss, C., Schneider, F.: Coordinated multi-robot exploration. IEEE Transactions on Robotics 21(3), 376–386 (2005)
2. Hayes, A.T.: How many robots? Group size and efficiency in collective search tasks. In: Proceedings of the 6th Int. Symp. on Distributed Autonomous Robotic Systems, DARS 2002, pp. 289–298 (2002)
3. Howard, A., Mataric, M.J., Sukhatme, G.S.: An incremental self-deployment algorithm for mobile sensor networks. Autonomous Robots 13(2), 113–126 (2002)
4. Mei, Y., Lu, Y.H., Hu, Y., Lee, C.: Deployment of mobile robots with energy and timing constraints. IEEE Transactions on Robotics 22(3), 507–522 (2006)
5. Meier, D., Stachniss, C., Burgard, W.: Cooperative exploration with multiple robots using low bandwidth communication. In: Beyerer, J., Puente, F., Sommer, K. (eds.) Informationsfusion in der Mess- und Sensortechnik, pp. 145–157 (2006)
6. Roberts, J., Stirling, T., Zufferey, J.C., Floreano, D.: 2.5D infrared range and bearing system for collective robotics. In: Proceedings of the International Conference on Intelligent Robots and Systems, pp. 3659–3664. IEEE, Piscataway (2009)
7. Roberts, J., Zufferey, J.C., Floreano, D.: Energy management for indoor hovering robots. In: Proc. IROS 2008, pp. 1242–1247. IEEE, Piscataway (2008)
8. Rosenfeld, A., Kaminka, G.A., Kraus, S.: A Study of Scalability Properties in Robotic Teams. In: Coordination of Large-Scale Multiagent Systems, Part 1, pp. 27–51. Springer, Berlin (2006)
9. Rybski, P., Larson, A., Lindahl, M., Gini, M.: Performance evaluation of multiple robots in a search and retrieval task. In: Proceedings of the Workshop on Artificial Intelligence and Manufacturing, pp. 153–160. AAAI Press, Menlo Park (1998)
10. Stirling, T., Wischmann, S., Floreano, D.: Energy-efficient indoor search by swarms of simulated flying robots without global information. Swarm Intelligence 4(2), 117–143 (2010)
11. Zlot, R.M., Stentz, A., Dias, M.B., Thayer, S.: Multi-robot exploration controlled by a market economy. In: IEEE International Conference on Robotics and Automation, vol. 3, pp. 3016–3023 (May 2002)

Genetic Encoding of Robot Metamorphosis: How to Evolve a Glider with a Genetic Regulatory Network

Anne C. van Rossum

Almende B.V., Rotterdam, The Netherlands
anne@almende.com

In REPLICATOR [2][1] powerful reconfigurable robots are designed and constructed. Reconfigurable robots can dock together and form robot organisms. Robot organisms have the ability to morph from snakes into spiders, chairs, swarms, wheels. The problem we are facing is: *How to evolve self-organized robot metamorphosis*? The metamorphosis graph $A = \{S, T\}$ is a tuple of S, the set of all possible robot module configurations, and T the set of all transitions between those configurations. A configuration $s \in S, s = \{R, D\}$ consists out of R robots with D connections. If $D = \oslash$, s denotes a swarm. A fitness function f for reciprocal metamorphosis defines a maximum for a specific cycle in A. A metamorphic fitness function is used that defines maximum fitness for a dynamic body form called a *glider*: a snake growing its head, and losing its tail $a_g \in A$. The evolutionary search process needs to find a self-organized solution for a glider in A.

Gene Regulatory Networks for Metamorphosis

To evolve metamorphosis a transition is needed from a static DNA string to a series of body configuration transitions. Estvez and Lipson call the genome a dynamical blueprint [1]. Quick et al. [3] demonstrated *post-developmental* dynamics using a gene regulatory network (GRN). To implement metamorphosis on modular robots we will add differentiated multi-cellularity and three-dimensional spatial encoding. Let the artificial genome be G. Transcription is defined as $tr : G \xrightarrow{tr} R$. Each $r \in R$ is an input-output device that takes as input a certain protein $p_i \in P$ and as output another protein $p_k \in P$. A genetic regulatory network N is a tuple $\{R, W\}$. W describes the wiring between the Rs. The evolutionary process can be depicted as $E = \{N, M\}$, where M denotes transitions (mutations) from one N to another. G is mapped via N to a robot configuration by protein quantities that are translated into docking and undocking events: $d : p \in P \xrightarrow{d} t \in T$. A gene $g = \{p_{in}, p_{out}, p_{low}, p_{high}, p_\delta\}$, say, $g_{example} = \{4, 8, 38, 83, 17\}$ is activated when protein with id 4 is between 38 and 83 (mole) entities, and increases the amount of protein 8 with 17 (mole)

[1] An FP7 project "Robotic Evolutionary Self-Programming and Self-Assembling Organisms".

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 564–565, 2010.

elements. Each robot module contains a virtual 3D cellular grid. Each grid cell runs the same GRN and its state is defined by the protein vector $\boldsymbol{P}_{cell}$. The existence of a docking connection, say between $M_i[c_{u,v,w}]$ and $M_j[c_{x,y,z}]$ induces cross-robot module diffusion. This is an epigenetic information stream (eco-devo besides just evo-devo) needed for module differentiation.

Simulation of a Glider

The gene regulatory network for self-organised metamorphosis is implemented in the Symbricator3D simulator.[2] A glider can indeed be evolved. The simulation to evolve a glider of 3 modules starting with an (evolved) snake of 8 modules requires on average (10 runs) 100 generations.[3] Larger gliders have not been found in a reasonable time! Interesting are the constraints that allow us to actually evolve this glider: 1.) Protein quantities are set to 50 (mole) entities. Randomization would decrease evolutionary convergence. 2.) A staged fitness function is needed to reward not just $a_g \in A$, but also $a_s \in A$: the formation of a snake to guide the evolutionary search. 3.) An individual organism's docking preferences are matched against the glider pattern, rather than a swarm evolution in which robot modules and organisms in all possible different states coexist.

This simulation shows that a gene regulatory network is expressive enough to act as a dynamic blueprint for a morphing robot organism. Further studies need to investigate the metamorphic landscape. This landscape might not have a smooth fitness function (like covered distance or height with locomotion or block climbing). The search for the proper mapping from a network of interacting genes to a network of interacting robot modules appears to be non-trivial.

Acknowledgements. The "REPLICATOR" project is funded by the European Commission within the work programme "Cognitive Systems, Interaction, Robotics" under the grant agreement no. 216240.

References

[1] Estévez, N., Lipson, H.: Dynamical blueprints: exploiting levels of system-environment interaction. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, p. 244. ACM, New York (2007)

[2] Kernbach, S., Hamann, H., Stradner, J., Thenius, R., Schmickl, T., van Rossum, A.C., Sebag, M., Bredeche, N., Yao, Y., Baele, G., de Peer, Y.V., Timmis, J., Mohktar, M., Tyrrell, A., Eiben, A.E., McKibbin, S.P., Liu, W., Winfield, A.F.T.: On adaptive self-organization in artificial robot organisms. In: Proc. of the First IEEE International Conference on Adaptive and Self-adaptive Systems and Applications (IEEE ADAPTIVE 2009), Athens/Glyfada, Greece (2009)

[3] Quick, T., Nehaniv, C., Dautenhahn, K., Roberts, G.: Evolving embodied genetic regulatory network-driven control systems. LNCS, pp. 266–277. Springer, Heidelberg (2003)

[2] The Symbricator3D simulator is built on top of the open-source Delta3D simulator.

[3] A movie of an evolved glider can be seen on `http://replicator.almende.com`

How Ant Systems Can Help in Management of pH for Industrial Wastewater Discharges

Marta Verdaguer[1], Jordi Giró[1], Narcís Clara[1], and Manel Poch[2]

[1] University of Girona, Catalonia, Spain
[2] The Catalan Institute for Water Research, Catalonia, Spain
{marta.verdaguer,narcis.clara,manuel.poch}@udg.edu,
jordiii.giro@gmail.com

Many processes from chemical industries generate wastewater discharges with acidic or alkaline character.These type of discharges often need a neutralization process before their incorporation into the receiving media. This process is complex: it presents many difficulties due to the none-linear response of pH value to the addition of acids or bases. Moreover, it requires considering the variable buffering capacity of system and the changes in loading characteristics [1,2]. The mixture of wastewater discharges usually has an unknown value of buffer capacity and its pH value can not be calculated easily.

In this work we approach the pH management for a complex network system of industrial wastewater discharges with an adaptation of ant system algorithm [3,4]. Let us consider a system of n industries $\{I_i\}$ with available volumes $\{v_i\}$ in their retention tanks. Each industry can discharge in the same basin different possible volumes N_i^j with $j \in \{1, \ldots, v_i\}$. Wastewater discharges can present different pH_i values and different buffering capacities β_i which value can be calculated in a previous step. The value β_i is assigned to 1 when the industrial wastewater has not buffering capacity. Considering N as the available hydraulic capacity for industrial wastewater discharges and B as the maximum expected value of pH, the most favorable value of pH corresponds to the maximization of the function Z defined by:

$$Z = -\log\left(\frac{\sum_{i=1}^{n}(\sum_{j=1}^{v_i} x_i^j N_i^j)10^{-\mathrm{pH}_i}}{\sum_{i=1}^{n}\beta_i(\sum_{j=1}^{v_i} x_i^j N_i^j)}\right) \quad . \tag{1}$$

Z is constrained by the restrictions $\sum_{j=1}^{v_i} x_i^j = 1$, $\sum_{i=1}^{n}\sum_{j=1}^{v_i} x_i^j N_i^j \leq N$ and $Z \leq B$ where x_i^j are the decision variables with $x_i^j \in \{0, 1\}$ for any $j = 1, \ldots, v_i$.

The implemented ant system algorithm works with two-stage process[3]. The first stage corresponds to the main phase in which the solutions are constructed. The second phase is aimed to improve the constructed solutions. The process has been applied to one scenario that comprises several industrial wastewater discharges defined as follows: 76% of acid character, 20% of alkaline character and 4% of them near to neutrality. Moreover, 52% have components with buffering

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 566–567, 2010.

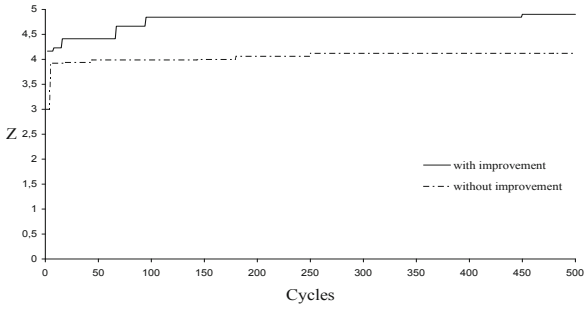


Fig. 1. Evolution of the best solution with and without improvement

capacity and a 48% have not it. To solve the transition rule of the main phase the following values for the testing parameters have been used: $a = 2, 8$; $b = 2, 8$; $\alpha = 0.2, 0.5, 0.8, 1, 2, 3, 4$ and $\beta = 0.2, 0.5, 0.8, 1, 2, 3, 4$. Each combination has been made with $\rho = 0.99$, $Q = 1000$, 500 cycles and 10 algorithm repetitions [5]. We have obtained the best average solution $\bar{Z} = 4.654$ with $a = 2$, $b = 2$, $\alpha = 0.5$ and $\beta = 0.2$. Figure 1 describes the evolution of the best solution with improvement $Z = 4.903$ and without improvement $Z = 4.117$.

As future work we plan to perform a more complete evaluation of the results with other algorithms related to ant system and with the simulation of other real scenarios.

References

1. Adroer, M., Alsina, A., Aumatell, J., Poch, M.: Wastewater neutralization control based on fuzzy logic: experimental results. Industrial and Engineering Chemistry Research 38, 2709–2719 (1999)
2. Garrido, M., Adroer, M., Poch, M.: Wastewater neutralization control based in fuzzy logic: simulation results. Industrial and Engineering Chemical Research 36, 1665–1674 (1997)
3. Dorigo, M., Stützle, T.: Ant colony optimization. MIT Press, Cambridge (2004)
4. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics B 26, 29–41 (1996)
5. Nahas, N., Nourelfath, M.: Ant system for reliability optimization of a series system with multiple-choice and budget constraints. Reliability Engineering and System Safety 87, 1–12 (2005)

Hybrid Metaheuristic Combining Ant Colony Optimization and *H*-Method

Leonid Hulianytskyi and Sergii Sirenko

V.M. Glushkov Institute of Cybernetics NAS Ukraine, Kyiv, Ukraine
leonhul.icyb@gmail.com, ssirenko@ieee.org

The paper presents a hybrid metaheuristic method ACO-H [4] for combinatorial optimization problems that combines two population-based approaches – ant colony optimization (ACO) and H-method. ACO is a multi-agent metaheuristic that has been successfully applied to many difficult optimization problems [1]. H-method is an extension of the discrete downhill simplex method [6] that applies during the search process specially defined segments. Similar to H-method ideas were introduced in the context of tabu search and now are known as path relinking [2].

Suggested hybrid metaheuristic is aimed to preserve good qualities both of used basic approaches. H-method is able to exploit good solutions quickly, but has a disadvantage of premature convergence. Randomized solution construction performed in ACO can be used to reinitialize H-method population in a more convenient manner than simple restart strategy. At the same time H-method can serve as a sophisticated solution improver for the ACO algorithm.

In the ACO-H metaheuristic H-method is integrated into the ACO algorithm and can be considered as a part of daemon actions, like widely used local search. One of the hybridization features is that the H-method algorithm is not run in each iteration of the ACO algorithm. There are special conditions determining when to transfer the control from ACO to H-method.

ACO-H algorithm first initializes the ACO algorithm. Iteration of hybrid algorithm starts with solution construction by ants. Then an optional pheromone update is performed. After that a control transfer condition is checked. If it is satisfied the iteration of ACO is interrupted and the H-method algorithm is started. If not, execution of the ACO algorithm iteration is continued.

We suggest two modifications of the hybrid algorithm differing in the way the search experience is transferred from ACO to H-method. First, solutions constructed by ants on the last iteration can be directly used as an initial population for H-method. In this case H-method is switched in the ACO-H algorithm between two states – normal run and trial run. The trial run is characterized by weaker H-method stopping criterion (it performs less number of iterations). This protects H-method from performing long useless runs and enables ACO part of the hybrid metaheuristic to explore the search space more efficient.

In the second modification initial population for the H-method algorithm is formed by sampling solutions that are close (in a metric sense) to the ACO algorithm current iteration-best solution. A number of oriented metric segments [6] are drawn from the iteration-best solution. The minimal-value (maximal in case

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 568–569, 2010.

of maximization problem) solutions are chosen from the parts of the segments that are distant enough (minimal closeness of the solutions is controlled by an adaptive parameter) from the iteration-best solution. These solutions constitute starting population for the H-method algorithm. Such sampling scheme provides population of solutions that are located in some "good" region of the search space. Adaptive increase of the minimal distance between iteration-best and sampled solutions enables ACO-H to leave basins of local optima attraction.

When the H-method algorithm converges, execution of the ACO algorithm iteration is continued. If the H-method algorithm run was performed, then its last iteration population is used to update pheromone values instead of solutions generated by ants. Iteration of the hybrid algorithm ends with daemon actions, if any. Suggested metaheuristic do not have any specific termination conditions, so general stopping criteria such as execution time limit can be used.

Two developed modifications were implemented using MMAS [1] and H-method algorithms and compared with independent basic algorithms results. Experiments were performed on symmetric traveling salesman problem instances from TSPLIB and on multidimensional assignment problem instances with known unique optimum provided by generator suggested in [3].

The results of the experimental evaluation on TSP and MAP instances show that suggested combining scheme can increase performance of the basic algorithms, but suffers from the lack of scalability. Being the best performing on TSP instances with less than 1000 cities, hybrid algorithms perform statistically the same as independent MMAS on the larger problems. Developing adaptive techniques for increased scalability or merging of two developed modifications is necessary in order to overcome this disadvantage.

Another future direction of our work is studying other types of metaheuristics hybridization [5].

Acknowledgements. We are grateful to the anonymous reviewers for their helpful comments.

References

1. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
2. Glover, F.: A template for scatter search and path relinking. In: Hao, J.K., Lutton, E., Ronald, E.M.A., Schoenauer, M., Snyers, D. (eds.) AE 1997. LNCS, vol. 1363, pp. 3–54. Springer, Heidelberg (1998)
3. Grundel, D.A., Pardalos, P.M.: Test problem generator for the multidimensional assignment problem. Comput. Optim. Appl. 30(2), 133–146 (2005)
4. Hulianytskyi, L., Sirenko, S.: Combining ant colony optimization and H-method (in russian). In: Markov, K., Ivanova., K., Mitov., I. (eds.) Decision Making and Business Intelligence Strategies and Techniques, pp. 95–102. FOI ITHEA, Sofia (2008)
5. Hulianytskyi, L., Sirenko, S.: Cooperative model-based metaheuristics. To appear in Electronic Notes in Discrete Mathematics (2010)
6. Hulianytskyi, L.: Deformation method in discrete optimization (in russian). Issled. Oper. ASU 34(2), 30–33 (1989)

Increasing Individual Density Reduces Extra-Variance in Swarm Intelligence

Ryusuke Fujisawa[1], Shigeto Dobata[2], and Fumitoshi Matsuno[3]

[1] Department of Mechanical Engineering,
Hachinohe Institute of Technology, Aomori, Japan
swarm.ant@gmail.com
[2] Department of Environmental Sciences and Technology,
Faculty of Agriculture, University of the Ryukyus, Okinawa, Japan
dobatan@gmail.com
[3] Department of Mechanical Engineering and Science,
Graduate School of Engineering, Kyoto University, Kyoto, Japan
matsuno@me.kyoto-u.ac.jp

Social organisms form a swarm and forage preys, collectively and effectively [1]. The swarm has to inhibit a variance of foraging frequency for survival, which ensures stable and predictable income. In the previous study, we focused on "trail pheromone" system which enables robots to communicate one another [2]. In the present study, we analysed statistically the variance of the foraging behaviour of the robot swarm, using repeated (48 trials) computer simulation. We set the individual density as the parameter. The density in Figures, X axis means the number of individuals on the unit field(180×180 [cm]). In the simulation, we set 360×360[cm] as the field size. When the individual density is low, the variance of the foraging behaviour is larger than expected from the random behaviour(i.e., binomial distribution; Fig. 1). Horizontal lines in Fig. 1 mean expected one from the binomial distribution. As the density goes high, the variance is reduced toward the expected value.

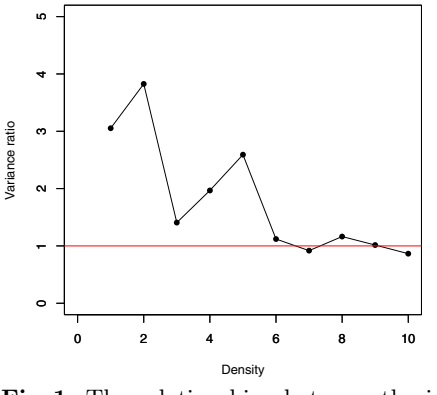


Fig. 1. The relationships between the individual density and the relative variance of (LD+RI) frequencies; X axis is the individual density. Y axis is the ratio of the variances.

We further analysed the result by separating the foraging behaviour into "laying down" and "reinforcing" behaviours. The laying down (LD) is a behaviour by which the robot find the prey individually, and the reinforcing (RI) is a behaviour by which it follows the pheromone trail and reaches the prey. Both behaviours are followed by the pheromone-laying behaviour. The observed pattern of variance reduction was different between laying down and reinforcing behaviours, and the variance reduction of total behaviour was mainly determined by the latter (Fig. 2).

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 570–571, 2010.

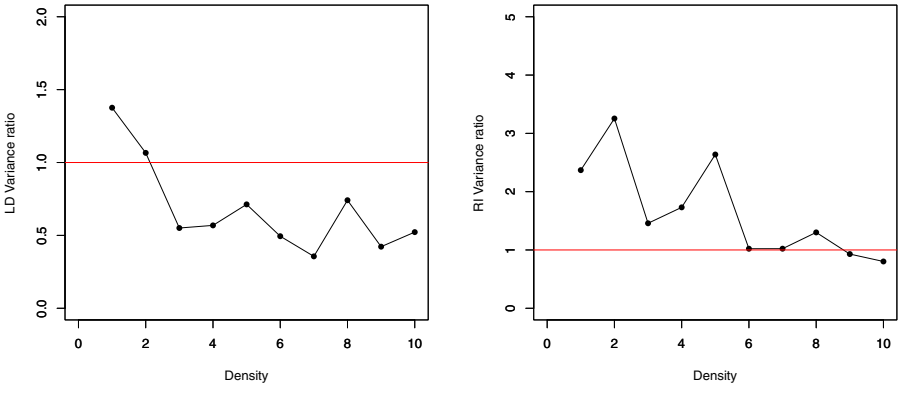


Fig. 2. The relationships between the individual density and the relative variance of LD frequencies (left) and RI frequencies (right); X axis is the individual density. Y axis is the ratio of the variances.

These results can be understood in the context of the stable existence of pheromone trail on the field. Pheromone-communicating systems harbor two sources of variance: one arises from the inevitable property of swarms (i.e., randomness) and governs the variance of LD, and the other arises from the unstable existence of the pheromone and governs the variance of RI. The larger group size may be adaptive in reducing the uncertainty in the existence of trail pheromone, which contributes to the reduction of the RI variance. The previous study [3] ascribed the mechanism of variance reduction in group-living to the law of large numbers, which holds true when the individual number is relatively small. Our pheromone-communicating system suggests that the larger group size is adaptive in reducing the uncertainty in the existence of trail pheromone. Because the underlying mechanism is based on the individual density, this can be applied when the individual number is rather large.

References

1. Wilson, E.O.: Sociobiology: The new synthesis. Belknap Press of Harvard University Press, Cambridge (1975)
2. Fujisawa, R., Dobata, S., Kubota, D., Imamura, H., Matsuno, F.: Dependency by concentration of pheromone trail for multiple robots. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) ANTS 2008. LNCS, vol. 5217, pp. 283–290. Springer, Heidelberg (2008)
3. Wenzel, J.W., Pickering, J.: Cooperative foraging, productivity, and the central limit theorem. Proceedings of the National Academy of Sciences of the United States of America 88, 36–38 (1991)

"Look out!": Socially-Mediated Obstacle Avoidance in Collective Transport

Eliseo Ferrante, Manuele Brambilla, Mauro Birattari, and Marco Dorigo

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{eferrant,mbrambil,mbiro,mdorigo}@ulb.ac.be

In collective transport, a group of robots has to cooperate in order to transport an object. Collective transport is necessary when transporting the object is hard or impossible for a single robot. The task is particularly difficult when communication bandwidth is limited, there is no access to global information or when using a decentralized approach. In these cases, an effective distributed coordination among the robots is necessary.

In the task we studied, a group of robots have to transport an object to a goal location, while avoiding obstacles along the way. The existing literature considers only either collective transport to a goal location in obstacle-free environments [1,3] or collective transport in a random direction within a cluttered environment [4].

In our study, three identical robots attach to an irregularly shaped object and collectively transport it from an initial to a goal location. The study was performed entirely in simulation. The robots we used are modeled after the foot-bot robot, in development for the Swarmanoid project[1]. The environment in which the robots move is an arena where a number of cuboid obstacles are present, each with an arbitrary position and orientation. A light source, with high intensity so that it can be perceived by all the robots, is placed in the environment. The presence of obstacles and the need to move to a given goal location create the need for handling conflicting individual decisions which can be caused by the non uniform perception of the environment.

We implemented a behavior composed of two sub-behaviors: *social mediation* and *collective transport*. The *social mediation* behavior is used to obtain a heading direction, mediated through all the transporting robots, to be used for the collective transport behavior. This heading direction has to take into account, at a given time, the presence or absence of obstacles and the goal direction. Once this socially mediated heading direction is obtained, it is used by the *collective transport* behavior to perform collective transport by setting the correct actuators' output.

The idea behind the social mediation behavior is the following. A robot's internal state can assume two different values: S_{social} or $S_{stubborn}$. When a robot possesses the information about the goal direction or when it perceives an obstacle, its state is set to $S_{stubborn}$. In this state, the robot computes the correct angle of motion (for example the angle for moving towards the goal while avoiding obstacles) and sends this to its neighbors. When a robot is completely uninformed

[1] http://www.swarmanoid.org

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 572–573, 2010.

(it does perceive neither the goal nor the obstacles), its state is set to S_{social}. In this state, the robot acts as a repeater, that is, it computes the average of the messages received by its neighbors, denoted with θ_S, and sends this value to its neighbors. The main idea is that the opinion of the stubborn robots can diffuse in the entire group thanks to the social inviduals. A motion control rule is then used to achieve motion, which uses θ_S as target direction to be followed.

We performed experiments in an arena where an obstacle is positioned at the center. We varied the angle α between the obstacle and the angle perpendicular to the direction of motion. Eight different arenas with different α were used: 0, $\pm 30°$, $\pm 45°$, $\pm 60°$, $90°$. For each arena we executed 100 runs. Results shows that the more α tends to 0, the longer it takes to avoid the obstacle. In all the runs, robots successfully reached the goal location.

We tested the proposed behavior also in another arena, in which obstacles were located at random positions with random orientations. In this scenario, the robots were able to reach the goal 96% of the time without collisions with the obstacles.

A video showing a typical run for this set of experiments can be found in [2].

Acknowledgements. This work was supported by the *SWARMANOID* project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission, under grant IST-022888, and by the *VIRTUAL SWARMANOID* project funded by the Fund for Scientific Research F.R.S.-FNRS of Belgium's French Community. The information provided is the sole responsibility of the authors and does not reflect the European Commission's opinion. The European Commission is not responsible for any use that might be made of data appearing in this publication. M. Dorigo and M. Birattari acknowledge support from the F.R.S.-FNRS of Belgium's French Community, of which they are a research director and a research associate, respectively.

References

1. Campo, A., Nouyan, S., Birattari, M., Groß, R., Dorigo, M.: Negotiation of goal direction for cooperative transport. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 191–202. Springer, Heidelberg (2006)
2. Ferrante, E., Brambilla, M., Birattari, M., Dorigo, M.: Look-out!: Socially mediated obstacle avoidance in collective transport: Complete data (2010), Supplementary information page at, http://iridia.ulb.ac.be/supp/IridiaSupp2010-005/
3. Groß, R., Dorigo, M.: Towards group transport by swarms of robots. International Journal of Bio-Inspired Computation 1(1-2), 1–13 (2009)
4. Trianni, V., Dorigo, M.: Self-organisation and communication in groups of simulated and physical robots. Biological Cybernetics 95, 213–231 (2006)

On Possible Connections between Ant Algorithms and Random Matrix Theory

Carlo Mastroianni

ICAR-CNR, Rende(CS), Italy
mastroianni@icar.cnr.it

This paper reports on a conjecture concerning the statistical behavior of Self-Chord [1], a self-organizing P2P system in which the resource keys are dynamically sorted with an ant algorithm. In Self-Chord (http://self-chord.icar.cnr.it), peers are organized in a logical ring, as in Chord, and a hash function is used to assign an index to every peer, and an access key to every resource. Contrary to Chord though, the values of resource keys are decoupled from those of peer indexes, and are dynamically sorted by ant-inspired agents through statistical *pick* and *drop* operations. This allows Self-Chord to keep the Chord capacity for serving discovery requests in logarithmic time, but leads to many further advantages, among which the possibility of assigning a semantic meaning to keys, a better load balancing among peers, and the efficient execution of range queries.

Figure 1 reports a sample snapshot of a Self-Chord network, in which peer indexes and resource keys are defined over 6 and 3 bits, respectively. At the interior of the ring, the figure specifies the indexes of the peers, whereas at the exterior it reports, for every peer, some of the keys stored by the peer, and the peer centroid. The latter is defined as the real value that minimizes the average distance between itself and the keys stored by the peer. Both the values of centroids and peer indexes are sorted in clockwise direction, but they are not related to one another. Indeed, different approaches are used to sort them: the peer indexes are sorted by Chord-like management operations, whereas the keys are dynamically sorted by the operations of the Self-Chord agents.

Interestingly, it emerged that the statistical distribution of the peer centroids is very similar to the distribution of the eigenvalues of random matrices taken from the GUE, Gaussian Unitary Ensemble. These matrices are used to model a wide class of complex dynamical systems, especially in the domain of nuclear physics [3]. The GUE matrices are also the subject of the Montgomery-Odlyzko law, which states that the distribution of the spacings between the non-trivial zeros of the Riemann Zeta function is statistically identical to the spacings of GUE eigenvalues. Figure 2 reports a comparison between the theoretical distribution of GUE spacings and the distribution of centroid spacings in Self-Chord networks with a number of peers N_p equal to 2000, 5000 and 10000. The two distributions are very similar, and the similarity increases with the size of the network. This observation is also supported by several qualitative considerations on the similarity between the behavior of Self-Chord centroids and that of the energy levels of physical systems modeled by GUE matrices [2].

The Self-Chord algorithm is very similar to many ant-inspired sorting algorithms (see the book on Swarm Intelligence authored by Bonabeau et al.).

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 574–575, 2010.

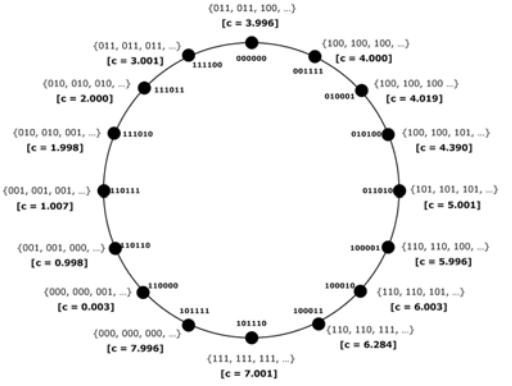


Fig. 1. Sample Self-Chord network. For each peer, its index, a number of stored keys and the centroid are reported.

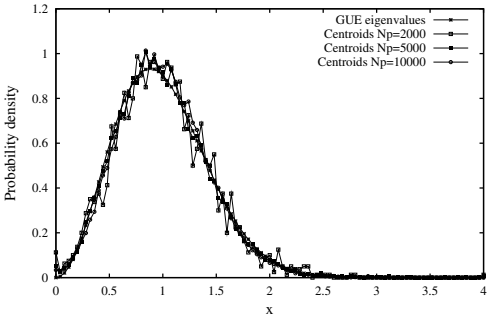


Fig. 2. Comparison between the distribution of spacings between consecutive centroids in Self-Chord and the theoretical distribution of spacings between GUE eigenvalues

Therefore, it is plausible that this similarity, if confirmed, could apply to ant algorithms in general, not only to Self-Chord. This would suggest the hypothesis that the mathematical nature of ant algorithms is inherently connected to random matrix theory and, more widely, to number theory. It is the opinion of the author that this fascinating conjecture is worth being analyzed more deeply and with more rigorous tests. More information can be found in [2].

References

1. Forestiero, A., Leonardi, E., Mastroianni, C., Meo, M.: Self-Chord: a bio-inspired P2P framework for self-organizing distributed systems. IEEE/ACM Transactions on Networking (2010)
2. Mastroianni, C.: A statistical analysis of Self-Chord: on possible connections between ant algorithms and random matrix theory. Tech. Rep. RT-ICAR-CS-10-02, ICAR-CNR, (May 2010), http://www.icar.cnr.it/tr/2010/02
3. Mehta, M.L.: Random Matrices. Academic Press Inc., Boston (1991)

Soft Variable Fixing in Path Relinking: An Application to ACO Codes

Antonio Bolufé Röhler[2], Marco A. Boschetti[1], and Vittorio Maniezzo[1]

[1] University of Bologna, Bologna, Italy
[2] University of Habana, Habana, Cuba
vittorio.maniezzo@unibo.it

Soft variable fixing has emerged as one of the main techniques that the area of *matheuristics* can contribute to general metaheuristics. Recent years have in fact witnessed a fruitful interplay of methods that were originally proposed as general metaheuristcs with methods rooted in mathematic programming, which can be applied alone or as hybrids for solving combinatorial optimization problems. In this work, we show how one of the most effective matheuristics techniques, namely soft variable fixing, can be hybridized with Ant Colony Optimization. Specifically, we will combine a standard ACO code with a path relinking operator, implemented by means of soft variable fixing.

Soft Variable Fixing is an operation based on Local Branching. This last is a technique originally introduced by Fischetti and Lodi [4], which works as follows. Starting from a feasible *reference solution* $\bar{x}$ of a mixed integer problem, the objective is to derive an exact exploration of a suitable neighborhood, defined on the binary representation of the reference solution. Following a positive integer parameter k, a *k-OPT neighborhood* $N(\bar{x}, k)$ of $\bar{x}$ is defined as the set of the feasible solutions satisfying an additional *local branching constraint*. This is a constraint that counts the number of variables which change their value, and limits their number to be at most k. This permits to have the best solution which differs from $\bar{x}$ in at most k positions.

Soft variable fixing has a wider scope than local branching, as it does not need to start with a full reference solution, but it can start with whichever subset of variables one need to fix and implement the local branching strategy in such a way as to ensure the feasibility of the optimized solutions.

The overall idea of our work is to let the ACO explore the search space, keeping a pool of the best solutions encountered. Each time a new solution enters the pool, path relinking is performed toward other pool solutions. Path relinking is delegated to the use of a Mixed Integer Programming (MIP) solver. This can be actually done in different ways [3], here we instantiate a local branching [4] on a seed solution obtained by a combination of the decision variable values of the solutions we are relinking. As opposed to similar approaches, we allow to modify up to a bounded number of variables which are common to both endpoint solutions, hence the "softness" of the variable fixing.

The problem chosen for validating the hybrid ACO algorithm is the 2 Dimensional Strip Packing problem (2SP), which is a problem encountered when cutting a set of rectangular pieces from a single rectangular strip, a stock sheet of width W, minimizing the waste. This problem instance is defined by a list of

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 576–577, 2010.

n rectangles, each having a width no greater than W. The rectangles must be packed into the semi-infinite strip, with their sides parallel to the strip edges, so that they do not overlap each other or the edges of the strip. the objective is to minimize the length of the packing is minimized.

The main structure of the proposed solution is that of the ANTS implementation [5] of the ACO framework, with an extension maintaining a pool —P— of the n_p best solutions found during search, to provide the basis for path relinking.

The elements to define for the 2SP-specific algorithm are solution representation, trail initialization, visibility computation, and the bound to use for trail update. In our case the solutions, evolved by the ants, were 0-1 vectors which concatenated variables representing the relative positions of the different pieces (above/below, left/right). Given these, a LP solver takes hundredths of a second to generate feasible coordinates for the pieces in the solution.

The bound was defined as the height reached after adding to the partial solution an area equal to the sum of the areas of the still to be placed items. This is not really a lower bound in the general case, but it is in our code given the particular construction heuristic we use.

Visibility was then computed as a combination of area, perimeter and bound, preferring the pieces with greater area, longer perimeter and yielding a lower bound. Trails were initially null.

Path relinking is triggered each time a new solution enters (and possibly one exits) pool P. The new solution is projected toward every other solution in P, in turns. During each projection, only the binary variables are considered. The subset of common ones is fixed and the problem is accordingly optimized. Soft fixing is instantiated by allowing a suitable number of otherwise fixed variables to be changed. The actual number derives from the computation of the number of 0/1 variables that need to be changed in order to allow 1, 2 or 3 pieces to be moved in the solution.

So far, we have implemented the above described procedure in $c\#$, using CoinMP [1] as a MIP solver. We run our tests on a standard benchmark from the literature, using the test set originally proposed by Berkey and Wang [2] for the 2-dimensional bin packing problem and setting the bin height to infinite. Computational results are still preliminary but encouraging, and they prove that this new methodology can effectively complement ant colony codes when applied to combinatorial optimization problems.

References

1. Coinmp project page (2010), https://projects.coin-or.org/CoinMP
2. Berkey, J., Wang, P.: Two dimensional finite bin packing algorithms. J. Oper. Res. Soc. 38, 423–429 (1987)
3. Boschetti, M., Maniezzo, V.: Combining exact methods and heuristics. In: Encyclopedia of Operations Research and Management Science. Wiley, Chichester (2010) (to appear)
4. Fischetti, M., Lodi, A.: Local branching. Math. Program. B 98, 23–47 (2003)
5. Maniezzo, V.: Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. INFORMS Journal on Computing 11(4), 358–369 (1999)

Training Randomly Connected, Recurrent Artificial Neural Networks Using PSO

Vytautas Jancauskas

Department of Computer Science, Faculty of Mathematics and Informatics,
Vilnius University, Vilnius, Lithuania
Vytautas.Jancauskas@mif.stud.vu.lt

The basic particle swarm algorithm was described by Kennedy and Eberhart [4]. In this paper a modified method with time varying inertia coefficient [3] was used where the inertia coefficient w goes linearly from w_{start} to w_{end}.

An approximated gradient descent algorithm was implemented so that it would be possible to compare the efficiency of particle swarm optimizer to a method that is well understood. Say we have a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ and want to find a minimum point $\boldsymbol{x}$. Lets call initial solution $\boldsymbol{x_0}$. For i-th solution $\boldsymbol{x_i}$ we approximate gradient $\boldsymbol{g_i}$, where coordinate $g_{ij} = (f(\boldsymbol{x_{i\Delta j}}) - f(\boldsymbol{x_i}))/\delta$ Here, vector $\boldsymbol{x_{i\Delta j}}$ is $\boldsymbol{x_i}$ with j-th coordinate set to $x_{ij} + \delta$. We create new solution $\boldsymbol{x_{i+1}} = \boldsymbol{x_i} - \gamma \times \boldsymbol{g_i}$ and repeat the whole process over again. Coefficient γ is called the learning rate and in this implementation goes from γ_{start} to γ_{end}. Gradient descent was chosen because it is the basis of most artificial neural network (ANN) training methods.

The ANN that is proposed in this paper is randomly connected and recurrent. Connections between individual computing units are established randomly with probability p and no limitations are placed on them. There are input, output and hidden units. Input units are initialized with the input vector during the first iteration. Output units contain the output of the network after all the calculations are done. And all units that don't fall in one of those two categories are called hidden. Suppose a network has n_x input neurons, n_h hidden neurons and n_y output neurons. Synaptic weight connecting units i and j is w_{ij}, $w_{ij} \neq 0$ with probability p. Network works in iterations. During the first iteration we set the activation value of each input neuron with an appropriate value in the input vector. After a specified number of iterations output neurons contain the output of the network. We define matrix $\boldsymbol{W}$ to contain all synaptic weights, where w_{ij} connects i-th computing unit to j-th computing unit. We define function $f(\boldsymbol{x}) = 1/(1 + e^{-2\boldsymbol{x}})$ which operates on vectors. Then for the i-th iteration we calculate the output as $\boldsymbol{x_{i+1}} = f(\boldsymbol{x_i} \times \boldsymbol{W})$, where $\boldsymbol{x_0}$ is a vector in which the first n_x coordinates are set to the values in the input vector and the following $n - n_x$ coordinates are set to 0.

It was shown how we can use this type of network for classification and time-series prediction. In the first case we optimize a function that takes as input a set of synaptic weights and computes squared error sum for a data set. And in case of time-series prediction we set the inputs of the network with a moving window over some time series and optimize the squared difference between actual

M. Dorigo et al. (Eds.): ANTS 2010, LNCS 6234, pp. 578–579, 2010.

network output and the data sample right after the window. That is we optimize the function that tells us how well does our network predicts the next sample in the time series. Both these applications were tested. Iris data set [6] was used for classification and sun spot data [1], specifically [2], was used for time series prediction. Also a method similar to the one used by Meissner et al. [5] was used in an attempt to find a good set of swarm and network parameters to classify the Iris data set. A set of parameters derived was iterations = 3, hidden units = 2, initialization range = (-0.32, 0.32), $v_{max} = 3.03$, $c_1 = 1.54$, $c_2 = 2.69$, $w_{start} = 1.25$ and $w_{end} = -0.16$. For gradient descent parameters $\gamma_{start} = 0.01$ and $\gamma_{end} = 0.0001$, starting and ending values for learning rate, were chosen after experimentaly verifying that they give good results.

Iters	Hidden	Conn	Dim	STrain	SContr	SMis	GTrain	GContr	GMis
2	0	0.4	20	22.162	24.571	45.2	24.645	23.8	25.0
2	0	0.6	29	9.228	14.188	18.5	8.297	10.169	12.9
2	0	0.8	39	5.215	12.37	13.5	4.117	6.617	5.4
2	2	0.4	32	8.507	15.002	14.1	21.564	23.184	30.1
2	2	0.6	49	3.871	10.07	9.6	2.740	6.322	4.7
2	2	0.8	65	2.7	9.591	6.7	2.111	6.017	4.0

The meanings of abbreviations are: Iters - number of network iterations, Hidden - number of hidden neurons, Conn - network connectivity, Dim - dimensionality of the solution space, STrain - swarm training error, SContr - swarm control error, SMis - misclassified samples in the control set when training with swarm, GTrain - gradient descent training error, GContr - gradient descent control error, GMis - misclassified samples in the control set when training with gradient descent.

It can be seen that we are able to use particle swarms to train neural networks that aren't well understood but which may have practical use. Gradient descent seems to give better performance overall, however efficient gradient descent algorithms are not known for most real argument function minimization problems. Further changes to the network can be made to improve it, such as including activation function shape in to the optimized function or automatic connection pruning.

References

1. http://www.ngdc.noaa.gov/stp/solar
2. ftp://ftp.ngdc.noaa.gov/STP/SOLAR%5FDATA/SUNSPOT%5FNUMBERS/INTERNATIONAL/monthly/MONTHLY.PLT
3. Eberhart, R.C., Shi, Y.: Parameter selection in particle swarm optimization. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 591–600. Springer, Heidelberg (1998)
4. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
5. Meissner, M., Schmuker, M., Schneider, G.: Optimized particle swarm optimization (opso) and its application to artificial neural network training. BMC Bioinformatics 7, 125 (2006)
6. Fisher, R.A.: Iris data set, http://archive.ics.uci.edu/ml/datasets/Iris

Author Index

Abdelbar, Ashraf M. 167
Absil, Pierre-Antoine 13
Al-Badarneh, Amer F. 464
Alba, Enrique 227
Altshuler, Yaniv 536
Álvarez, Víctor 368
Amato, Paolo 408
Armario, José Andrés 368
Azzam-ul-Asar, 275

Baiboun, Nadir 287
Bailis, Peter 263
Bari, Md. Faizul 312
Băutu, Andrei 512
Beal, Jacob 179
Bengoetxea, Endika 416
Berro, Alain 60
Bertinat, María Elisa 336
Bharat, Tadikonda Venkata 448
Birattari, Mauro 203, 239, 251, 287, 572
Bogon, Tjorben 554
Borckmans, Pierre B. 13
Borrotti, Matteo 352
Boschetti, Marco A. 576
Bouamama, Salim 464
Boukerram, Abdellah 464
Bourjot, Christine 344
Brambilla, Manuele 572
Bruckstein, Alfred M. 36, 119, 215, 536
Brutschy, Arne 287
Butler, Matthew 504

Cerofolini, Gianfranco 408
Charpillet, François 344
Charrier, Rodolphe 344
Christensen, Anders Lyhne 400, 558
Clara, Narcís 566
Coello Coello, Carlos A. 48
Cole, Jason 72
Cordón, Oscar 472
Crailsheim, Karl 84, 424

Dahl, Torbjørn S. 24
Davison, Timothy 1
De Lucrezia, Davide 352
Di Caro, Gianni A. 456
Dinh, Huy Q. 360
Dixon, Clare 440
Dobata, Shigeto 570
Dorigo, Marco 251, 287, 400, 558, 572
Ducatelle, Frederick 456

Elor, Yotam 36, 119, 215
Engelbrecht, Andries P. 191, 560
Escario, Jose B. 552

Farooq, Muddassar 392
Fernández-Martínez, Juan L. 496
Ferrante, Eliseo 251, 572
Fisher, Michael 440
Floreano, Dario 562
Förster, Alexander 456
Frau, María Dolores 368
Frison, Marco 287
Fujimoto, Noriyuki 488
Fujisawa, Ryusuke 570

Gambardella, Luca 456
García-Gonzalo, Esperanza 496
Ghosh, Ashish 376
Ghosh, Susmita 376
Giró, Jordi 566
Giron-Sierra, Jose M. 552
Gudiel, Félix 368
Güemes, Belén 368
Guinand, Frédéric 520
Günther, Maik 384

Halder, Anindya 376
Helvik, Bjarne E. 480
Huan, Hoang Xuan 360
Hulianytskyi, Leonid 568

Iqbal, Shahrear 312
Ishteva, Mariya 13

Jacob, Christian 1
Jancauskas, Vytautas 578
Jimenez, Juan F. 552
Johnson, Colin G. 528
Jourdan, Laetitia 227

Kazakov, Dimitar 504
Khan, Affan 275
Khouadjia, Mostepha Redouane 227
Konur, Savas 440
Korb, Oliver 72
Kötzing, Timo 324
Kouno, Asuki 556

Larrañaga, Pedro 416
Lattner, Andreas D. 554
Leguizamón, Guillermo 48
Li, Ke 299
Liu, Wenguo 107
López-Ibáñez, Manuel 95
Luchian, Henri 512

Maniezzo, Vittorio 576
Marie-Sainte, Souad Larabi 60
Martín, Elena 368
Masserini, Massimo 408
Mastroianni, Carlo 574
Mathews, Nithin 251, 400
Matsuno, Fumitoshi 570
Mauri, Giancarlo 408
Mayet, Ralf 84
Minervini, Giovanni 352
Minh, Bui Quang 360
Miorandi, Daniele 143
Moeslinger, Christoph 424
Montes de Oca, Marco A. 203, 251
Moraglio, Alberto 528
Mukerji, Tapan 496

Nagpal, Radhika 263
Neumann, Frank 324
Nissen, Volker 384
Nolfi, Stefano 155

O'Grady, Rehan 400, 558
Osuna, Amparo 368
Otero, Fernando E.B. 528
Owen, Jennifer 432

Padula, Darío 336
Paquereau, Laurent 480
Pellegrini, Paola 239
Phillips, David 1
Pigné, Yoann 520
Pinciroli, Carlo 558
Pini, Giovanni 287
Poch, Manel 566
Poli, Irene 352
Poursanidis, Georgios 554

Quirin, Arnaud 472

Rahman, M. Sohel 312
Riaz-ul-Hasnain, 275
Roberz, Jonathan 84
Robledo, Franco 336
Rodríguez-Bocca, Pablo 336
Röglin, Heiko 324
Röhler, Antonio Bolufé 576
Roli, Andrea 287
Romero, Pablo 336
Romero-Zaliz, Rocío 472
Rossi, Louis 299
Ruiz-Gazen, Anne 60

Sadeequllah, Muhammad 275
Salama, Khalid M. 167
Saleem, Muhammad 392
Sarker, Md. Omar Faruque 24
Schmickl, Thomas 84, 424
Schmidt, Adam 544
Schoeman, Isabella 560
Sharma, Jitendra 448
Shen, Chien-Chung 299
Sirenko, Sergii 568
Sperati, Valerio 155
Stepney, Susan 432
Stirling, Timothy 562
Stützle, Thomas 95, 203, 239
Suzuki, Einoshin 556

Takano, Shigeru 556
Talbi, El-Ghazali 227
Thomas, Kyle 299
Timm, Ingo J. 554
Timmis, Jonathan 432
Torres, Claudio 299
Tran, Nam-Luc 287
Trianni, Vito 155
Tsutsui, Shigeyoshi 488

Ullah, Israr 392

van Rossum, Anne C. 564
Verdaguer, Marta 566
von Haeseler, Arndt 360
von Mammen, Sebastian 1

Werfel, Justin 131, 263
Winfield, Alan F.T. 107, 432
Witt, Carsten 324

Yamamoto, Lidia 143
Yuan, Zhi 203