

Operations Research/Computer Science Interfaces Series

Lionel Amodeo
El-Ghazali Talbi
Farouk Yalaoui *Editors*

Recent Developments in Metaheuristics



 Springer

Operations Research/Computer Science Interfaces Series

Volume 62

Series Editors:

Ramesh Sharda

Oklahoma State University, Stillwater, Oklahoma, USA

Stefan Voß

University of Hamburg, Hamburg, Germany

More information about this series at <http://www.springer.com/series/6375>

Lionel Amodeo • El-Ghazali Talbi
Farouk Yalaoui
Editors

Recent Developments in Metaheuristics

 Springer

Editors

Lionel Amodeo
Laboratory of Industrial Systems
Optimization
University of Technology of Troyes
Troyes, France

El-Ghazali Talbi
INRIA Laboratory
CRISTAL/CNRS
University of Lille 1
Villeneuve d'Ascq, France

Farouk Yalaoui
Laboratory of Industrial Systems
Optimization
University of Technology of Troyes
Troyes, France

ISSN 1387-666X

Operations Research/Computer Science Interfaces Series

ISBN 978-3-319-58252-8

ISBN 978-3-319-58253-5 (eBook)

DOI 10.1007/978-3-319-58253-5

Library of Congress Control Number: 2017948033

© Springer International Publishing AG 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature

The registered company is Springer International Publishing AG

The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This book aims to give the reader some of the most recent research dealing with the development of metaheuristics.

The document gathers 28 chapters. These chapters could be divided into two main sets. The first one, Chaps. 1–10, is dedicated specifically to present some new optimization and modeling techniques based on metaheuristics. The goal of the second set, Chaps. 11–28, is to develop some advanced metaheuristic approaches to solve real-life applications issue such as scheduling, vehicle routing problem, multimedia sensor network, supplier selection, bin packing, objects tracking, radio frequency identification.

All the results proposed in the present document were accepted and presented during the conferences MIC'15, the eleventh edition of the Metaheuristics International Conference, which was held from June 7 to 10, 2015, in Agadir, Morocco, and META'14, the fourth edition of the International Conference on Metaheuristics and Nature Inspired Computing, which was held from October 27 to 31, 2014 in Marrakech, Morocco.

The first chapter, entitled “Hidden Markov Model Classifier for the Adaptive Particle Swarm Optimization,” by Oussama Aoun, Malek Sarhani, and Abdellatif El Afia, presents an integration of hidden Markov Model (HMM) particle swarm optimization (HMM) in APSO (adaptive particle swarm optimization) to have a stochastic state classification at each iteration. To tackle the problem of the dynamic environment during iterations, an additional online learning for HMM parameters is integrated into the algorithm using online expectation-maximization algorithm. The authors performed evaluations on ten benchmark functions to test the HMM integration inside APSO.

The second chapter, by Oumayma Bahri, Nahla Ben Amor, and El-Ghazali Talbi, is dedicated to deal with the possibilistic framework for multi-objective optimization under uncertainty. This chapter addresses the multi-objective problems with fuzzy data, in particular, with triangular-valued objective functions. To solve

such problems, the authors have proposed an extension of two multi-objective evolutionary algorithms SPEA2 and NSGA-II, by integrating a new triangular Pareto dominance.

The third chapter, “Combining Neighborhoods into Local Search Strategies,” by Renaud De Landtsheer, Yoann Guyot, Gustavo Ospina, and Christophe Ponsard, develops a declarative framework for defining local search procedures. It proceeds by combining neighborhoods by means of so-called combinators that specify when neighborhoods should be explored and introduce other aspects of the search procedures such as stop criteria, solution management, and various metaheuristics. The approach proposed by the authors introduces these higher-level concepts natively in local search frameworks in contrast with the current practice which still often relies on their adhoc implementation in imperative language.

The fourth chapter, “All-Terrain Tabu Search Approaches for Production Management Problems,” by Nicolas Zufferey, Jean Respen, and Simon Thevenin, is dedicated to the presentation of tabu search approaches with enhanced exploration and exploitation mechanisms. For this purpose, some specific ingredients are discussed: different neighborhood structures (i.e., different types of moves), guided restarts based on a distance function, and deconstruction/reconstruction techniques.

The fifth chapter, “A Re-characterization of Hyper-heuristics,” by Jerry Swan, Patrick De Causmaecker, Simon Martin, and Ender Özcan, tackles with hyper-heuristic optimization methodology. Hyper-heuristic search has traditionally been divided into two layers: a lower problem-domain layer (where domain-specific heuristics are applied) and an upper hyper-heuristic layer (where heuristics are selected or generated). The interface between the two layers is commonly termed the “domain barrier”. The authors show how it is possible to make use of domain knowledge without loss of generality and describe generalized hyper-heuristics which can incorporate arbitrary domain knowledge.

The sixth chapter, “POSL: A Parallel-Oriented Metaheuristic-Based Solver Language,” by Alejandro Reyes Amaro, Eric Monfroy, and Florian Richoux, proposes a parallel-oriented solver language (POSL, pronounced “puzzle”), a new framework to build interconnected metaheuristic-based solvers working in parallel. The novelty of this approach lies in looking at solver as a set of components with specific goals, written in a parallel-oriented language based on operators. A major feature in POSL is the possibility to share not only information, but also behaviors, allowing solver modifications during runtime. POSL’s main advantage is to allow solver designers to quickly test different heuristics and parallel communication strategies to solve combinatorial optimization problems, which are usually time-consuming and very complex technically, requiring a lot of engineering.

The seventh chapter, “An Extended Neighborhood Vision for Hill-Climbing Move Strategy Design,” by Sara Tari, Matthieu Basseur, and Adrien Goëffon, aims at determining pivoting rules that allow hill-climbing to reach good local optima. The authors propose to use additional information provided by an extended neighborhood for an accurate selection of neighbors and introduce the maximum expansion pivoting rule which consists in selecting a solution which maximizes the improvement possibilities at the next step.

The eighth chapter, “Theory Driven Design of Efficient Genetic Algorithms for a Classical Graph Problem,” by Dogan Corus and Per Kristian Lehre, presents a principled way of designing a genetic algorithm which can guarantee a rigorously proven upper bound on its optimization time. The shortest path problem is selected to demonstrate how level-based analysis, a general-purpose analytical tool, can be used as a design guide. We show that level-based analysis can also ease the experimental burden of finding appropriate parameter settings.

The ninth chapter, “On the Impact of Representation and Algorithm Selection for Optimisation in Process Design: Motivating a Metaheuristic Framework,” by Eric S. Fraga, Abdellah Salhi, and El-Ghazali Talbi, aims at demonstrating that the method choice does matter. For a set of problems, all in the same domain of heat exchanger network synthesis, different combinations of method and representation work best for individual problems. This motivates the development of an overarching method which could identify the best combination and solve the problem most effectively. The authors propose a Multiple Heuristics, Multiple Representation (MHMR) paradigm which mirrors the Multiple Algorithm, Multiple Formulation (MAMF) model for the exact solution. Exploring this paradigm, say through the design and implementation of prototype software frameworks will be the focus for future work in our respective research groups.

The tenth chapter, “Manufacturing Cell Formation Problem Using Hybrid Cuckoo Search Algorithm,” by Bouchra Karoum, Bouazza Elbenani, Noussaima El Khattabi, and Abdelhakim A. El Imrani, presents an adapted optimization algorithm entitled the cuckoo search algorithm for solving this kind of problems. The proposed method is tested on different benchmark problems; the obtained results are then compared to others available in the literature.

Chapter 11, “Hybridization of Branch-and-Bound Algorithm with Metaheuristics for Designing Reliable Wireless Multimedia Sensor Network,” by Omer Ozkan, Murat Ermis, and Ilker Bekmezci, contributes to deploy sensor nodes to maximize the WMSN reliability under a given budget constraint by considering terrain and device specifications. The reliable WMSN design with deployment, connectivity, and coverage has NP-hard complexity, therefore a new hybridization of an exact algorithm with metaheuristics is proposed. A branch-and-bound approach is embedded into hybrid simulated annealing (HSA) and hybrid genetic algorithm (HGA) to orient the cameras exactly. Since the complexity of the network reliability problem is NP-complete, a Monte Carlo simulation is used to estimate the network reliability.

Chapter 12, “A Hybrid MCDM Approach for Supplier Selection with a Case Study,” by Hanane Asselaou, Brahim Ouhbi, and Bouchra Frikh, considers the supplier selection problem where one of the strategic decisions that have a significant impact on the performance of the supply chain. In this chapter, the supplier selection problem of a well-known refining company in Africa is investigated, and an integrated DEMATEL-ANP-TOPSIS methodology is used to select the best supplier providing the most customer satisfaction for the criteria determined.

Chapter 13, “A Multi-objective Optimization via Simulation Framework for Restructuring Traffic Networks,” subject to increases in population by Enrique Gabriel Baquela, and Ana Carolina Olivera, studies a nonlinear and stochastic problem

which is the traffic network design problem. The origin-destination traffic assignment problem is a particular case of this problem. The authors propose the use of a multi-objective particle swarm optimization together with traffic simulations in order to generate restructuring alternatives that optimize both, traffic flow and cost associated to this restructure.

Chapter 14, by S. Chaimatanan and D. Delahaye and M. Mongeau, deals with hybrid metaheuristic for air traffic management with uncertainty, the 4D trajectory optimization of each aircraft so as to minimize the probability of potential conflicts between trajectories. A hybrid-metaheuristic optimization algorithm has been developed to solve this large-scale mixed-variable optimization problem. The algorithm is implemented and tested with real air traffic data, taking into account uncertainty over the French airspace for which a conflict-free and robust 4D trajectory plan is produced.

Chapter 15, by Michaela Zehetner and Walter J. Gutjahr, considers the sampling-based genetic algorithms for the bi-objective stochastic covering tour problem. The authors presented different approaches for solving an extended version of the covering tour problem (CTP), namely, the bi-objective stochastic.

Chapter 16, “A Metaheuristic Framework for Dynamic Network Flow Problems,” by M. Hajjem, H. Bouziri, and E.G. Talbi, considers the definition of a metaheuristic framework for the NP-hard flow over time problems. A specific case study of dynamic flow problem is treated, precisely the evacuation problem from a building. Therefore, the authors have supposed that the dynamic maximum flow model with flow-dependent transit time could handle the dynamic property and the crowdedness on nodes and arcs. The genetic algorithm as a population-based evolutionary method to treat this NP-hard problem is proposed.

In Chap. 17, “A Greedy Randomized Adaptive Search for the Surveillance Patrol Vehicle Routing Problem,” by Simona Mancini, a new rich vehicle routing problem is introduced, the surveillance patrol vehicle routing problem (SPVRP). This problem came out from a real need of a surveillance company to create fairer routing plans for its security patrols. The problem consists into routing a set of patrols in order to visit a set of checkpoints. Each checkpoint requires one or more visits, each one of which is to be performed within a fixed time window. Minimum time spacing between two consecutive visits should be observed. The goal is to minimize cost while minimizing, at the same time, time windows and minimum spacing constraint violations. To address this problem, a greedy randomized adaptive search algorithm is used to provide good solutions, and a further GRASP algorithm is used to generate pools of good solutions.

Chapter 18, “Strip Algorithms as an Efficient Way to Initialize Population-Based Metaheuristics,” by Birsan İrem Selamoğlu, Abdellah Salhi, and Muhammad Sulaiman, presents the strip algorithm (SA) which is a constructive heuristic. This method has been tried on the Euclidean travelling salesman problem (TSP) and other planar network problems with some success. The authors set out to investigate new variants such as the 2-part strip algorithm (2-PSA), the spiral strip algorithm (SSA) and the adaptive strip algorithm (ASA).

Chapter 19, “Matheuristics for the Temporal Bin Packing Problem,” by Fabio Furini and Xueying Shen, develops an extension of the bin packing problem, where items consume the bin capacity during a time window only. The problem asks for finding the minimum number of bins to pack all the items respecting the bin capacity at any instant of time. Both a polynomial-size formulation and an extensive formulation are studied. Various heuristic algorithms are developed and compared, including greedy heuristics and a column generation-based heuristic.

Chapter 20, “A Fast Reoptimization Approach for the Dynamic Technician Routing and Scheduling Problem,” by V. Pillac, C. Guéret, and A.L. Medaglia, the technician routing and scheduling problem (TRSP) consists in routing staff to serve requests for service, taking into account time windows, skills, tools, and spare parts. The authors tackle the dynamic TRSP (D-TRSP) with new requests appear over time. They propose a fast reoptimization approach based on a parallel adaptive large neighborhood search (RpALNS) able to achieve state-of-the-art results on the dynamic vehicle routing problem with time windows. In addition, the authors solve a set of randomly generated D-TRSP instances and discuss the potential gains with respect to a heuristic modeling a human dispatcher solution.

Chapter 21, “Optimized Air Routes Connections for Real Hub Schedule Using SMP SO Algorithm,” by H. Rahil, B. Abou El Majd, and M. Bouchoum, presents study dealing with the choice to open new routes for air carriers, airports and regional governments have some tools to promote desirable connections to be offered toward specific destinations. With a given flight program, the air carrier decision to open new routes faces several constraints and affects the flight schedules in a remarkable way. This chapter is the first to introduce the problem of connectivity in the network of an airline whose main activity is based on air hub structure, optimizing the insertion of new airline routes while ensuring the best fill rate seats and avoiding significant delays during correspondence. Quality of service index (QSI) will be considered as a dual parameter for the profit of a newly opened market. The experimental tests are based on real instance of Royal Air Maroc flights schedule on the hub of Casablanca.

Chapter 22, “Solving the P/Prec;Cj/Cmax Using an Evolutionary Algorithm,” by Dalila Tayachi, tackles the problem of scheduling a set of related tasks on a set of identical processors, taking into account the communication delays with the objective of minimizing the maximal completion time. As the problem is well known as NP-hard, a particle swarm optimization (PSO) is proposed to solve it. The proposed approach HEA-LS is a hybrid algorithm involving particle swarm optimization (PSO) and local search algorithm (LSA). Experiments conducted on several benchmarks known in the literature prove the effectiveness of the proposed approach and show that it compares very well to the state-of-the-art methods.

Chapter 23, “A User Experiment on Interactive Reoptimization Using Iterated Local Search,” by David Meignan, presents an experimental study conducted with subjects on an interactive reoptimization method applied to a shift scheduling problem. The studied task is the adjustment, by a user, of candidate solutions provided by an optimization system in order to introduce a missing constraint. Two proce-

dures are compared on this task. The first one is a manual adjustment of solutions assisted by software that dynamically computes the cost of the current solution. The second procedure is based on reoptimization. For this procedure, the user defines some desired changes on a solution, and then a reoptimization method is applied to integrate the changes and re-optimize the rest of the solution.

Chapter 24, “Surrogate-Assisted Multi-objective Evolutionary Algorithm for Fuzzy Job Shop Problems,” by Juan José Palacios, Jorge Puente, and Camino R. Vela, Inés González-Rodríguez and El-Ghazali Talbi, considers a job shop scheduling problem with uncertain processing times modeled as triangular fuzzy numbers and propose a multi-objective surrogate-assisted evolutionary algorithm to optimize not only the schedules’ fuzzy makespan but also the robustness of schedules with respect to different perturbations in the durations. The surrogate model is defined to avoid evaluating the robustness measure for some individuals and estimate it instead based on the robustness values of neighboring individuals, where neighbor proximity is evaluated based on the similarity of fuzzy makespan values.

In Chap. 25, “Toward a Novel Reidentification Method Using Metaheuristics,” by Tarik Ljouad, Aouatif Amine, and Ayoub Al-Hamadi, and Mohammed Rziza, tracking multiple moving objects in a video sequence can be formulated as a profile matching problem. The authors introduce a novel modified cuckoo search (MCS) based reidentification algorithm. A complex descriptor representing each moving person is built from different low-level visual features such as the color and the texture components. The authors make use of a database that involves all previously detected descriptors, forming therefore a discrete search space where the sought solution is a descriptor and its quality is represented by its similarity to the query profile.

Chapter 26, “Facing the Feature Selection Problem with a Binary PSO-GSA Approach,” by Malek Sarhani, Abdellatif El Afia, and Rdouan Faizi, considers feature selection. The latter has become the focus of much research in many areas where we can face the problem of big data or complex relationship among features. Metaheuristics have gained much attention in solving many practical problems, including feature selection. The contribution of the authors is to propose a binary hybrid metaheuristic to minimize a fitness function representing a trade-off between the classification error of selecting the feature subset and the corresponding number of features. This algorithm combines particle swarm optimization (PSO) and gravitational search algorithm (GSA).

Chapter 27, “An Optimal Deployment of Readers for RFID Network Planning Using NSGA-II,” by Abdelkader Raghieb, Badr Abou El Majd, and Brahim Aghezzaf, considers radio frequency identification (RFID). RFID process depends on radio frequency waves to transfer data between a reader and an electronic tag attached to an item, in order to identify objects or persons, which allows an automated traceability. In order to optimize the deployment of RFID reader problem, the authors propose a new approach based on multi-level strategy using as main objectives the coverage, the number of deployed readers and the interference. Non-dominated sorting genetic algorithm II (NSGA-II) is adopted in order to minimize the total quantity of readers required to identify all tags in a given area.

Chapter 28, “An Enhanced Bat Echolocation Approach for Security Audit Trails Analysis Using Manhattan Distance,” by Guendouzi Wassila and Boukra Abdelmadjid, deals with the security audit trail analysis problem. This problem is classified as an NP-hard combinatorial optimization problem. The authors propose to use the bat echolocation approach to solve such a problem. The proposed approach, named an enhanced binary bat algorithm (EBBA), is an improvement of bat algorithm (BA). The fitness function is defined as the global attack risks.

Troyes, France
Villeneuve d’Ascq, France
Troyes, France
December 2016

Lionel Amodeo
El-Ghazali Talbi
Farouk Yalaoui

Contents

1	Hidden Markov Model Classifier for the Adaptive Particle Swarm Optimization	1
	Oussama Aoun, Malek Sarhani, and Abdellatif El Afa	
1.1	Introduction	1
1.2	Literature Review	2
1.3	Classification of APSO States by HMM	5
	1.3.1 Adaptive PSO Framework	5
	1.3.2 HMM Classification of Particle States	6
1.4	Experiment	10
	1.4.1 Parameters Setting	10
	1.4.2 Comparison on the Solution Accuracy	11
	1.4.3 Comparison on the Convergence Speed	12
1.5	Conclusion	14
	References	14
2	Possibilistic Framework for Multi-Objective Optimization Under Uncertainty	17
	Oumayma Bahri, Nahla Ben Amor, and El-Ghazali Talbi	
2.1	Introduction	17
2.2	Background on Deterministic Multi-Objective Optimization	18
2.3	Existing Approaches for Uncertain Multi-Objective Optimization	22
2.4	Proposed Possibilistic Framework for Multi-Objective Problems Under Uncertainty	23
	2.4.1 Basics on Possibility Theory	23
	2.4.2 Adaptation of Possibilistic Setting	26
	2.4.3 New Pareto Optimality over Triangular Fuzzy Numbers ..	27
	2.4.4 Extended Optimization Algorithm	31
2.5	Application on a Multi-Objective Vehicle Routing Problem	33
2.6	Conclusion	39
	References	40

3	Combining Neighborhoods into Local Search Strategies	43
	Renaud De Landtsheer, Yoann Guyot, Gustavo Ospina, and Christophe Ponsard	
3.1	Introduction	43
3.2	Related Work	45
3.3	Principle of Neighborhood Combinators	46
3.4	Six Shades of Warehouse Location	47
3.5	Building a Library of Combinators	51
3.5.1	Neighborhood and Move Selection Combinators	51
3.5.2	Acceptation Function Combinators	52
3.5.3	Solution Management Combinators	53
3.5.4	Stop Criterion Combinators	53
3.5.5	Code Embedding Combinators	54
3.5.6	Neighborhood Aggregation Combinators	54
3.6	A Vehicle Routing Example with Combinators	54
3.7	Conclusion	55
	References	57
4	All-Terrain Tabu Search Approaches for Production Management Problems	59
	Nicolas Zufferey, Jean Respen, and Simon Thevenin	
4.1	Introduction	59
4.2	Smoothing the Production for Car Sequencing	61
4.2.1	Presentation of the Problem	61
4.2.2	Solution Methods	62
4.2.3	Experiments	62
4.3	A Deconstruction-Reconstruction Method for Job Scheduling	63
4.3.1	Presentation of the Problem	63
4.3.2	Solution Methods	64
4.3.3	Experiments	65
4.4	Tabu Search with Diversity Control and Simulation	66
4.4.1	Presentation of the Problem	66
4.4.2	Solution Methods	67
4.4.3	Experiments	68
4.5	Dynamic Tabu Search for a Resource Allocation Problem	69
4.5.1	Presentation of Dynamic Tabu Search	69
4.5.2	Application to a Resource Allocation Problem	71
4.6	Conclusion	72
	References	72
5	A Re-characterization of Hyper-Heuristics	75
	Jerry Swan, Patrick De Causmaecker, Simon Martin, and Ender Özcan	
5.1	Introduction	76
5.1.1	Historical Development of Hyper-Heuristics	76
5.1.2	Effectiveness in New Domains	79
5.2	Popular Notion of the Domain Barrier	80

5.3	The Need for ‘Domain-Independent Domain Knowledge’	82
5.4	Cross-Domain Knowledge Representation	84
5.5	Future Directions: The Role of Ontologies	86
5.6	Conclusion	87
	References	87
6	POSL: A Parallel-Oriented Metaheuristic-Based Solver Language	91
	Alejandro REYES-Amaro, Eric Monfroy, and Florian Richoux	
6.1	Introduction	91
6.2	POSL Parallel Solvers	93
6.2.1	Operation Module	93
6.2.2	Open Channels	94
6.2.3	Computation Strategy	95
6.2.4	Solver Definition	99
6.2.5	Communication Definition	99
6.3	A POSL Solver	100
6.3.1	Connecting Solvers	102
6.4	Results	103
6.5	Conclusions	106
	References	107
7	An Extended Neighborhood Vision for Hill-Climbing Move Strategy Design	109
	Sara Tari, Matthieu Basseur, and Adrien Goëffon	
7.1	Introduction	109
7.2	Combinatorial Optimization, Local Search, Hill-Climbing	110
7.3	Hill-Climbing Moving Strategies: Description and Evaluation	112
7.3.1	Context	112
7.3.2	Experimental Protocol	113
7.3.3	Maximum Expansion vs. \mathcal{A}_1 Vision Area Climbers	114
7.3.4	Maximum Expansion vs. \mathcal{A}_2 Vision Area Climbers	117
7.4	Maximum Expansion Sophistication: A Multiobjectivized Approach	119
7.4.1	Multiobjectivization	120
7.4.2	Biobjectivized Pivoting Rules	121
7.5	Discussion	122
	References	123
8	Theory Driven Design of Efficient Genetic Algorithms for a Classical Graph Problem	125
	Dogan Corus and Per Kristian Lehre	
8.1	Introduction	125
8.2	Analysis of EAs on Shortest Path Problems	127
8.3	Method: Level-Based Analysis	129
8.4	Design of a Genetic Algorithm	131
8.4.1	Representation of Solutions	132

8.4.2	The Objective Function and Level Structure	132
8.4.3	The Mutation Operator	134
8.4.4	The Crossover Operator	135
8.4.5	Other Parameter Settings	136
8.4.6	All-Pairs Shortest Path Problem	136
8.5	Expected Running Time	137
8.6	Conclusion	138
	References	139
9	On the Impact of Representation and Algorithm Selection for Optimisation in Process Design: Motivating a Meta-Heuristic Framework	141
	Eric S. Fraga, Abdellah Salhi, and El-Ghazali Talbi	
9.1	Introduction	142
9.2	Representation	142
9.3	Motivating Example	143
9.4	The Match-Making Problem	145
9.5	Heat Exchanger Network Design	145
9.6	Representations and Algorithms for HENS	146
9.7	Results	147
9.8	Conclusions	148
	References	149
10	Manufacturing Cell Formation Problem Using Hybrid Cuckoo Search Algorithm	151
	Bouchra Karoum, Bouazza Elbenani, Noussaima El Khattabi, and Abdelhakim A. El Imrani	
10.1	Introduction	151
10.2	Problem Formulation	153
10.3	Improved Cuckoo Search Algorithm	154
	10.3.1 Basic Cuckoo Search	154
	10.3.2 The Proposed Cuckoo Search Algorithm	155
10.4	Computational Results	158
10.5	Conclusion	161
	References	161
11	Hybridization of Branch and Bound Algorithm with Metaheuristics for Designing Reliable Wireless Multimedia Sensor Network	163
	Omer Ozkan, Murat Ermis, and Ilker Bekmezci	
11.1	Introduction	163
11.2	The Problem Definition	165
11.3	Hybridization of Branch and Bound Algorithm with Metaheuristics	169
	11.3.1 Hybridization of Branch and Bound with Simulated Annealing	170

11.3.2	Hybridization of Branch and Bound with Genetic Algorithm	171
11.4	Experimental Study	172
11.4.1	Parameter Tuning	174
11.4.2	Performance Results	174
11.5	Conclusion	177
	References	177
12	A Hybrid MCDM Approach for Supplier Selection with a Case Study	179
	Hanane Assellaou, Brahim Ouhbi, and Bouchra Frikh	
12.1	Introduction	179
12.2	Related Works	181
12.3	Methodology	183
12.3.1	DEMATEL Method	184
12.3.2	ANP Method	185
12.3.3	TOPSIS Method	187
12.4	Proposed Methodology and Application Case	188
12.4.1	Identification of Necessary Criteria for Supplier Selection	189
12.4.2	Applying DEMATEL for Constructing the Interdependence Relationship Network	190
12.4.3	The Weights of Criteria Calculation	191
12.4.4	Application of TOPSIS in Alternatives Ranking	193
12.5	Conclusion	195
	References	196
13	A Multi-Objective Optimization via Simulation Framework for Restructuring Traffic Networks Subject to Increases in Population ..	199
	Enrique Gabriel Baquela and Ana Carolina Olivera	
13.1	Introduction	199
13.1.1	Literature Review	200
13.2	Origin-Destiny Traffic Assignment Problem	201
13.2.1	Traffic Systems	201
13.2.2	Origin-Destiny Traffic Assignment Problem	203
13.3	Traffic Simulations	204
13.4	Multiobjective Particle Swarm Optimization	205
13.4.1	Particle Swarm Optimization	205
13.4.2	Multi-Objective Particle Swarm Optimization	206
13.5	Optimization Framework	207
13.5.1	General Procedure	207
13.5.2	Solution Evaluation	208
13.6	Experiments	209
13.6.1	Scenarios	209
13.6.2	Tests	210
13.6.3	Algorithm Parameters	211

- 13.7 Results 211
 - 13.7.1 Convergence 211
 - 13.7.2 Uniformity 213
 - 13.7.3 Evolution of the Solutions Generated by the Algorithm .. 213
 - 13.7.4 Comparison with NSGA-II Metaheuristic 215
- 13.8 Conclusions 215
- References 216

- 14 Hybrid Metaheuristic for Air Traffic Management with Uncertainty** 219

S. Chaimatanan, D. Delahaye, and M. Mongeau

 - 14.1 Introduction 219
 - 14.1.1 Air Traffic Management: A Brief Review 220
 - 14.1.2 Strategic Aircraft Trajectory Planning 222
 - 14.2 Previous Related Works 224
 - 14.3 Mathematical Model 226
 - 14.3.1 Uncertainty 227
 - 14.3.2 Trajectory Separation Methods 229
 - 14.3.3 Optimization Formulation 232
 - 14.4 Interaction Detection Module 237
 - 14.5 Hybrid-Metaheuristic for Strategic Trajectory Planning 240
 - 14.5.1 Simulated Annealing: A Brief Overview 241
 - 14.5.2 Iterative-Improvement Local Search: A Brief Overview .. 242
 - 14.5.3 Hybrid Simulated-Annealing/Iterative-Improvement Local Search 243
 - 14.5.4 Neighborhood Function 246
 - 14.6 Computational Results 246
 - 14.7 Conclusions 248
 - References 250

- 15 Sampling-Based Genetic Algorithms for the Bi-Objective Stochastic Covering Tour Problem** 253

Michaela Zehetner and Walter J. Gutjahr

 - 15.1 Introduction 253
 - 15.2 The Bi-Objective Stochastic Covering Tour Problem 255
 - 15.2.1 The Covering Tour Problem 255
 - 15.2.2 The Bi-Objective Stochastic Extension 255
 - 15.2.3 Literature Review 257
 - 15.2.4 Mathematical Formulation 258
 - 15.3 Algorithms 261
 - 15.3.1 NSGA-II 262
 - 15.3.2 Savings Algorithm 265
 - 15.3.3 Sampling Procedures 265

15.4	Case Study	268
15.4.1	Implementation Details	268
15.4.2	Performance Measures	269
15.4.3	Test Instances	270
15.4.4	Computational Results	271
15.5	Conclusions	278
	References	281
16	A Metaheuristic Framework for Dynamic Network Flow Problems	285
	M. Hajjem, H. Bouziri, and El-Ghazali Talbi	
16.1	Introduction	285
16.2	Basic Notions and Results of Static Network Flow Problems	286
16.3	Dynamic Network	287
16.3.1	Time Horizon	287
16.3.2	Parameters	288
16.3.3	Representation	289
16.4	Flow Over Time Models	290
16.4.1	Maximum Dynamic Flow Problem	291
16.4.2	Earliest Arrival Flow Problem	291
16.4.3	Quickest Flow Problem	292
16.4.4	Dynamic Minimum Cost Flow Problem	292
16.4.5	Complexity of Dynamic Network Flow Problems	293
16.5	Metaheuristics	293
16.5.1	Blackboard-Based Metaheuristics	294
16.5.2	Evolutionary-Based Metaheuristics	294
16.6	An Evolutionary Framework for Dynamic Network Flow Problems	295
16.6.1	Solution Representation	295
16.6.2	Generation of Initial Solutions	295
16.6.3	Crossover Operator	295
16.6.4	Mutation Operator	296
16.7	Application to Evacuation Problem	296
16.7.1	A Case Study for Building Evacuation	297
16.7.2	Design of Genetic Algorithm	298
16.7.3	Results Analysis	300
16.8	Conclusion	302
	References	303
17	A Greedy Randomized Adaptive Search for the Surveillance Patrol Vehicle Routing Problem	305
	Simona Mancini	
17.1	Introduction	305
17.2	Literature Review	307
17.3	The Surveillance Patrols Vehicle Routing Problem	308

17.4	A GRASP for the Surveillance Patrol Vehicle Routing Problem . . .	309
17.4.1	Local Search	312
17.4.2	Feasibility Search	313
17.5	A GRASP for Solutions Pools Selection	313
17.6	Computational Tests	314
17.7	Conclusions and Future Developments	315
	References	316
18	Strip Algorithms as an Efficient Way to Initialise Population-Based Metaheuristics	319
	Birsen İrem Selamoğlu, Abdellah Salhi, and Muhammad Sulaiman	
18.1	Introduction	319
18.2	Ideas Behind the Strip Algorithm	320
18.2.1	The Appropriate Number of Strips	322
18.3	2-PSA: The 2-Part Strip Algorithm	323
18.4	Worst-Case Analysis of 2-PSA and an Upper Bound on the Minimum Tour Lengths Returned	323
18.5	Other Implementations of the Strip Algorithm	325
18.5.1	The Adaptive Strip Algorithm (ASA)	326
18.5.2	The Spiral Strip Algorithm (SSA)	326
18.6	Computational Results and Conclusion	327
	References	330
19	Matheuristics for the Temporal Bin Packing Problem	333
	Fabio Furini and Xueying Shen	
19.1	Introduction	333
19.1.1	Greedy-Type Algorithms	334
19.2	Compact Formulation	335
19.3	Extensive Formulation	338
19.3.1	Column Generation Heuristics	339
19.4	Computational Experiments	340
19.5	Conclusion	345
	References	345
20	A Fast Reoptimization Approach for the Dynamic Technician Routing and Scheduling Problem	347
	V. Pillac, C. Guéret, and A.L. Medaglia	
20.1	Introduction	348
20.2	Literature Review	349
20.3	The Parallel Adaptive Large Neighborhood Search	351
20.3.1	Destroy	352
20.3.2	Repair	354
20.3.3	Adaptive Layer	354
20.3.4	Objective Function	355
20.3.5	Acceptance Criterion	355

20.3.6	Computation of an Initial Solution	356
20.3.7	Solution Pool	356
20.4	Parallel Reoptimization Approach	356
20.5	Computational Results	358
20.5.1	Experimental Setting	358
20.5.2	Validation on the D-VRPTW	359
20.5.3	Results on the D-TRSP	361
20.6	Conclusions	365
	References	365
21	Optimized Air Routes Connections for Real Hub Schedule Using SMPSO Algorithm	369
	H. Rahil, B. Abou El Majd, and M. Bouchoum	
21.1	Introduction	369
21.2	Methods	372
21.2.1	Problem Formulation	372
21.2.2	SMPSO Algorithm	377
21.3	Results and Discussion	379
21.4	Conclusions and Future Works	382
	References	383
22	Solving the P/Prec, p_j, C_{ij}/C_{max} Using an Evolutionary Algorithm ...	385
	Dalila Tayachi	
22.1	Introduction	385
22.2	Problem Formalization	386
22.3	Related Work	387
22.4	A New Hybrid Evolutionary Algorithm Based on Particle Swarm Optimization HEA-LS	388
22.4.1	Particle Swarm Optimization	388
22.4.2	Position Representation and Fitness Evaluation	389
22.4.3	Position Update	390
22.4.4	The Hybrid HEA-LS Algorithm	391
22.5	Performance Comparison and Results	393
22.6	Conclusion and Perspectives	396
	References	397
23	A User Experiment on Interactive Reoptimization Using Iterated Local Search	399
	David Meignan	
23.1	Introduction	399
23.2	Interactive Reoptimization for Shift Scheduling	401
23.2.1	Interactive Process	401
23.2.2	Shift Scheduling	402
23.2.3	Optimization Procedures	404

23.3	Experiment	406
23.3.1	Objectives	406
23.3.2	Method	407
23.4	Results	410
23.5	Conclusion and Perspectives	412
	References	413
24	Surrogate-Assisted Multiobjective Evolutionary Algorithm for Fuzzy Job Shop Problems	415
	Juan José Palacios, Jorge Puente, Camino R. Vela, Inés González-Rodríguez, and El-Ghazali Talbi	
24.1	Introduction	415
24.2	Job Shop Scheduling with Uncertain Durations	417
24.2.1	Uncertain Durations	417
24.2.2	Robust Scheduling	418
24.2.3	The Multiobjective Approach	419
24.3	Multiobjective Evolutionary Algorithm	420
24.3.1	The Surrogate Model	420
24.3.2	Genetic Operators	422
24.4	Experimental Study	422
24.5	Conclusions	426
	References	427
25	Towards a Novel Reidentification Method Using Metaheuristics	429
	Tarik Ljouad, Aouatif Amine, Ayoub Al-Hamadi, and Mohammed Rziza	
25.1	Introduction	430
25.2	Object Representation	431
25.3	Projection of the Modified Cuckoo Search on the Multiple Object Tracking Problem	434
25.4	Experimental Results	439
25.5	Conclusion	443
	References	444
26	Facing the Feature Selection Problem with a Binary PSO-GSA Approach	447
	Malek Sarhani, Abdellatif El Afia, and Rdouan Faizi	
26.1	Introduction	447
26.2	Literature Review	449
26.3	The Proposed Binary PSOGSA Approach	450
26.3.1	The Canonical BPSO Algorithm	450
26.3.2	The Classical BGSA Algorithm	451
26.3.3	The Aggregative Multi-Objective Fitness Function of FS	452
26.3.4	The Proposed Hybrid Algorithm for FS	452

26.4	Experiment	455
26.4.1	Experiment Setup	455
26.4.2	Examination of BMPSOGSA for FS	456
26.4.3	Comparison with Well-Known FS Techniques	460
26.5	Conclusion	461
	References	461
27	An Optimal Deployment of Readers for RFID Network Planning Using NSGA-II	463
	Abdelkader Raghib, Badr Abou El Majd, and Brahim Aghezzaf	
27.1	Introduction	463
27.2	Problem Formulation	465
27.2.1	Number of Deployed Readers	466
27.2.2	Full Coverage	466
27.2.3	Interference	467
27.3	Methods	467
27.3.1	NSGA-II Algorithm	468
27.3.2	The Proposed Approaches	469
27.4	Results and Discussion	470
27.5	Conclusion	475
	References	475
28	An Enhanced Bat Echolocation Approach for Security Audit Trails Analysis Using Manhattan Distance	477
	Wassila Guendouzi and Abdelmajid Boukra	
28.1	Introduction	477
28.2	Related Work	478
28.3	Problem Formulation	479
28.4	Bat Algorithm Overview	480
28.5	Proposed Approach	481
28.5.1	Solution Representation	482
28.5.2	Initialization of Algorithm Parameters and Bat Population	482
28.5.3	Fitness Function and Manhattan Distance	482
28.5.4	Proposed Discretisation	484
28.5.5	Operators	485
28.6	Experimental Results	487
28.6.1	Performance Validation Using Random Data	488
28.6.2	Comparisons of EBBA with BBO, GA and HS Algorithms Using Real Data	490
28.7	Conclusion	491
	References	492
	Index	495

Chapter 1

Hidden Markov Model Classifier for the Adaptive Particle Swarm Optimization

Oussama Aoun, Malek Sarhani, and Abdellatif El Afia

Abstract Particle swarm optimization (PSO) is a stochastic algorithm based population that integrates social interactions of animals in nature. Adaptive Particle swarm optimization (APSO) as an amelioration of the original one, improve the performance of global search and gives better efficiency. The APSO defines four evolutionary states: exploration, exploitation, convergence, and jumping out. According to the state, the inertia weight and acceleration coefficients are controlled. In this paper, we integrate Hidden Markov Model Particle swarm optimization (HMM) in APSO to have a stochastic state classification at each iteration. Furthermore, to tackle the problem of the dynamic environment during iterations, an additional online learning for HMM parameters is integrated into the algorithm using online Expectation-Maximization algorithm. We performed evaluations on ten benchmark functions to test the HMM integration inside APSO. Experimental results show that our proposed scheme outperforms other PSO variants in major cases regarding solution accuracy and specially convergence speed.

Keywords Particle swarm optimization • Swarm intelligence • Hidden Markov model • Machine learning • Parameters adaptation • Metaheuristics control

1.1 Introduction

Nowadays, several approaches have been proposed to improve the performance and the convergence of particle swarm optimization (PSO) algorithm. One of the main challenges within PSO is to improve its capability of both global exploration and local exploitation abilities. To achieve these goals, a number of variants of PSO have been proposed. For example, PSO have been hybridized with several intelligent algorithms such as genetic algorithm, local search, artificial immune system and so on, in order to fix the problem of premature convergence. But, at the expense of rapid convergence. Therefore, it is important to control the PSO parameters to achieve the trade-off between the diversity and the convergence speed. That is, it is

O. Aoun • M. Sarhani (✉) • A. El Afia

Department of Informatics and Decision Support, Mohammed V University, Rabat, Morocco

e-mail: oussama.aoun@um5s.net.ma; malek.sarhani@um5s.net.ma;

a.elafia@um5s.net.ma

known that the accuracy of PSO depends on the selection of the appropriate values of parameters and their values through the search process. See for instance [15].

There are three main strategies which can be used to categorize PSO parameters, the first one is to affect this parameter randomly as a constant value, and the second one is that the parameter depends on the iteration number. In the third one, the value of this parameter at each iteration varies according to the results obtained by the particles until this iteration [34]. The third approach enables the PSO to be adaptive (adaptive PSO or APSO), another way to enhance the adaptivity of to PSO is to vary the population size [4]. This problem can be formulated as a learning process in which each particle learns from the obtained data and predict the values of the parameters in accordance with the history of its values and the values of other particles.

Several approaches have been used to improve PSO adaptivity including the APSO approach. In this paper, we use the probabilistic machine learning method by Hidden Markov Chain (HMM). That is, HMM is used to have stochastic state control of APSO at each iteration. The main idea is to assign state selection inside the adaptive particle swarm optimization to HMM. This process is performed by the Viterbi algorithm that gives the most probable path of states in each PSO iteration. Also, HMM parameters are calculated and updated at each iteration according to the change in particle environment. An online Expectation-Maximization algorithm gives a continuous parameter update for the HMM. At the best of our knowledge, the Hidden Markov Model (HMM) has not been used yet for this type of learning of PSO behavior.

The remainder of the paper is organized as follows. Section 1.2 provides a literature review. Section 1.3 describes the manner of integration of HMM in PSO. Section 1.4 describes the experimental results and Sect. 1.5 is dedicated to a conclusion.

1.2 Literature Review

In the last few years, the use of the learning concept has become promising to enhance PSO adaptivity and then to improve its performance. That is, various methods have been proposed to improve the learning capability of the particles in PSO. The learning process has been used in literature in different forms. van den Bergh and Engelbrecht [27] used the concept of cooperative learning which consists of using multiple swarms to optimize the various components of the solution vector cooperatively. This idea is similar to the multi-agent approach which consists on dividing the particles into agents. Another commonly used algorithm is the comprehensive learning [31]. In this case, each particle learns from another particle which is chosen according to a learning probability. The basic idea behind the orthogonal learning PSO proposed by Zhan et al. [33] is to determine the best combination of historical values of the particle itself and other particles. Another approach is the feedback learning which has been introduced by Tang et al. [26]. In the mentioned work, the feedback fitness information of each particle (described especially by each

particle's history best fitness) is used to determine the learning probabilities. These probabilities affect the acceleration parameters of PSO. In [9], the learning has been done by different examples instead of one. The convergence has been analyzed theoretically by considering a Markov process of the PSO algorithm.

In particular, some papers have adapted the learning concept to adjust PSO parameters. The parameters that have to be defined are the velocity clamping, the inertia weight (w) and the acceleration factors (cognitive attraction and social attraction). Thus, a number of methods have been proposed to learn the best values of these factors. For instance, Ding et al. [6] employed a stochastic local search to adjust the inertia weight in terms of keeping a balance between the diversity and the convergence speed. Also, Zhang et al. [34] proposed a Bayesian PSO in which the inertia weight vector is calculated by maximizing the posterior probability density function of the weight.

Furthermore, in [1], the inertia weight was also dynamically adjusted at each time step by taking into account the distance between the particles and gBest. The classical way to define the inertia weight was proposed by Shi and Eberhart [23], it consists on linearly decreasing w with the iterative generations. However, some papers proposed other variants of ω . For instance, Tang et al. [26] proposed a quadratic decreasing and Zhan et al. [32] chosen a sigmoid decreasing of this parameter. These time-varying method of the inertia weight is chosen in order to control the exploitation and exploration of PSO.

Concerning the acceleration parameters, according to [19], the particle swarm can be stable if the following conditions are satisfied:

$$\begin{aligned} 0 < c_1 + c_2 < 4 \\ (c_1 + c_2)/2 < w < 1 \end{aligned} \quad (1.1)$$

The most used adaptive strategy of c_1 and c_2 has been formulated by Zhan et al. [32], it consists in updating its values according to four defined states which are: exploration, exploitation, convergence and jumping out. Other papers interested in the relation between the learning behavior of PSO and acceleration parameters. For instance, Kamalapur and Patil [13] examined the link between these parameters and the topology of PSO. Moreover, Subbaraj et al. [24] interested in choosing the best values of these parameters.

Epitropakis et al. [7] studied the effect of the social and cognitive parameters on PSO convergence and used differential evolution to enhance it. In [10], the learning process of fitness information is used to control the parameters of PSO. Therefore, we can conclude that the learning ability of PSO is related to the choice of the acceleration parameters.

On the other hand, machine learning methods have gained much attention and wide application in several fields due to its ability of prediction with accurate precision. The configuration of metaheuristics by machine learning can be done in two ways. The off-line configuration involves the adjustment of parameters before running the algorithm while the online configuration consists on adjusting the algorithm parameters while solving the problem. In this paper, we interest on the online con-

figuration of metaheuristics. In particular, to learn and predict the PSO behavior, some papers applied machine learning for this task. For instance, a variant of reinforcement learning approach (learning automata) has been used in [8] to enhance the convergence of the comprehensive learning PSO. Also, the integration of classification algorithms for APSO has just been proposed by Liang et al. [16], their work consists on adjusting the inertia weight based on the evaluation results of the states of clusters and the swarm.

In terms of Hybridization between PSO and HMM, this has been done in most cases for the aim of improving HMM performance by PSO. Phon-Amnuaisuk [20] investigated the potential of the Particle Swarm Optimization (PSO) as an alternative method for HMM parameters instead of Baum-Welch algorithm. Furthermore, Sun and Liu [25] proposed hybrid algorithm combining Viterbi and PSO to exploit the randomness of HMMs parameters' training. Yang and Zhang [30] proposed a combination of Baum-Welch and PSO to train HMM parameters to deal with the time sequence classification problem. Moreover, some improvements of PSO have been proposed in some papers based on Markov chains. For instance, in [18], a definition of the state sequence of a single particle and the swarm state based on the proposed PSO difference model, then obtains the transition probability of the swarm transferring to the optimal set.

On the other hand, HMM is a statistical learning tool used for modeling generative sequences characterized by a set of observations [22]. HMM capacity is related to statistical learning and classification. This framework can be used to model stochastic processes where we have unobservable states of the system presented by a Markov process and observations related to states by probabilistic dependencies. HMM has been applied in many fields like speech recognition and engineering domains [22].

Considering Online HMM training, it was defined by Di Mauro et al. [5] and Polikar et al. [21], The main feature of an HMM which uses online learning is that it can independently learn from new block of data at a time. So, HMM parameters should be efficiently updated from new data without requiring access to all training data. In addition, parameters are re-estimated online upon observing each new subsequence [3]. The online EM algorithm for HMM allows continuous adaptation of HMM parameters along a potentially infinite observation stream.

In our approach, we use Online HMM at each iteration of the adaptive PSO (APSO) to classify particles states. This online classification of PSO particles enable to improve its performances by an online machine learning technique.

1.3 Classification of APSO States by HMM

1.3.1 Adaptive PSO Framework

First, PSO is a metaheuristic algorithm which was introduced by Kennedy and Eberhart [14]. The PSO concept consists of, at each time step, changing the velocity (accelerating) of each particle toward its $pBest$ and $gBest$ locations. Finally, the results are completed. Each particle i has two vectors: the velocity vector and the position vector according to Eqs. (1.2) and (1.3):

$$v_i = wv_i + c_1r_1(pBest - x_i) + c_2r_2(gBest - x_i) \quad (1.2)$$

$$x_i = x_i + v_i \quad (1.3)$$

Where: r_1 and r_2 are two independently uniformly distributed random variables in the interval $[0,1]$. w is the inertia weight; it varies linearly from 1 to near 0 during iterations. c_1 and c_2 are acceleration factors; c_1 is the cognitive component that measured the degree of self-confidence of a particle and measured the degree at which it trusts its performance. c_2 is the social component that relied in the capability of the swarm to find better candidate solutions.

Our approach is based on adaptive particle swarm optimization (APSO) which features better search efficiency than standard classical PSO. First, the APSO evaluate the population distribution and particle fitness, a real-time state estimation procedure is performed to identify one of four evolutionary states: exploration, exploitation, convergence, and jumping out. It permits to have automatic control of inertia weight, acceleration coefficients, and other PSO parameters. Then, an elitist learning strategy is performed when the evolutionary state is classified as convergence state. APSO improves either the search efficiency and convergence speed [32].

As regards the update of the inertia weight, it is done as follows in order to balance the global and local search capabilities as in the following equation:

$$\omega(f) = \frac{1}{1 + 1.5e^{-2.6f}} \in [0.4, 0.9] \forall f \in [0, 1] \quad (1.4)$$

The APSO gives the following major modifications:

- Evolutionary state estimation (ESE) technique and adaptive control according to the state.
- Elitist learning strategy (ELS): in the case of convergence state.

Special interest is given to ESE technique in our approach. In APSO, an evolutionary factor f is calculated at each iteration and state is given by a defuzzification technique [32]. Our method consists of replacing the defuzzification in APSO by online HMM classification as described in the next paragraph.

1.3.2 HMM Classification of Particle States

Our main contribution concerns the hybridization of HMM and APSO. That is, we use of classification and learning capabilities of HMM to enhance APSO. On the one hand, HMM is a stochastic method where we need to associate several probabilities in the model definition; transitions between states are governed by a set of probabilities named transition probabilities. In a particular state, observation is generated according to the related probability distribution. It is only the observation, not the state is visible that is why the state is hidden. After model parameters definition, a resolution algorithm is used to build HMM classification process.

We follow a similar approach than the one proposed by Zhan et al. [32] which is also used by Ardizzon et al. [1] to enhance PSO adaptivity. It consists of identifying one of four evolutionary states: exploration, exploitation, convergence, and jumping out in order to enable the automatic control of acceleration coefficients. Our contribution to these works is to use HMM to identify the proper state (class) at each iteration. So, we can generate the Markov Chain as described in Fig. 1.1. PSO parameters are updated according to the classified state at every iteration. Then, the same as Zhan et al. [32], an elitist learning strategy is performed when the evolutionary state is classified as a convergence state.

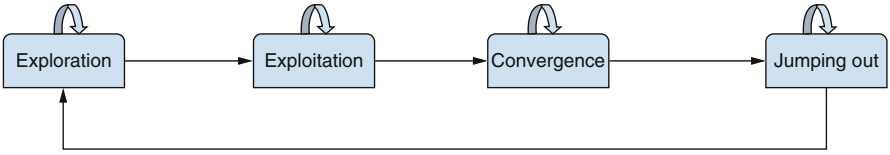


Fig. 1.1 Markov chain of APSO states

1.3.2.1 HMM Definition

We define the Hidden Markov Model by a triple $\lambda = (\Pi, A, B)$, all processes are defined on a probability space (Ω, F, P) :

- $\Pi = (\pi_i)$ The vector of the initial state probabilities;
- $A = (a_{ij})$ The state transition matrix, $P(X_t = i | X_{t-1} = j), i, j \in [1, N]$;
- $B = (b_{jk})$ The emission matrix also called the confusion matrix, $P(Y_t = k | X_t = j), j \in [0, N], k \in [0, M]$.

The state $\{X_t\}_{t \in N}$ takes values from the set $S = \{s_i\}_{i \in [1, 4]}$ what references respectively: exploration, exploitation, convergence, and jumping out. The change of state is reflected by the PSO sequence $s_1 \Rightarrow s_2 \Rightarrow s_3 \Rightarrow s_4 \Rightarrow s_1 \dots$, as deduced by Zhan et al. [32], corresponding to the Markov Chain in Fig. 1.1.

Furthermore, we define corresponding initial transition probabilities as:

$$P(X_t = i \vee X_{t-1} = j), i, j \in [1, 4].$$

This probability controls all behavior of transition between states of APSO resolution. We take for all possible i and j transitions as mentioned in Fig. 1.1 a probability of 0.5.

The initial state probability corresponds to deterministic start in exploration state:

$$\Pi = (\pi_i) = [1 \ 0 \ 0 \ 0] \quad (1.5)$$

The observed parameter of this hidden chain is the evolutionary factor f of the APSO. Observation will be belonging f to subintervals of $[0,1]$ ($[0,0.2]$, $[0.2,0.3]$, $[0.3,0.4]$, $[0.4,0.6]$, $[0.6,0.7]$, $[0.7,0.8]$, $[0.8,1]$). We divide $[0,1]$ to seven subintervals as mentioned by Zhan et al. [32], so the observation will be number of subintervals witch belong f . Initial observation probabilities are deduced from the difuzzification process defined in by Zhan et al. [32], we define coefficients values in intervals as emissions and we dress the matrix of membership corresponding to the difuzzification. It represents the emission probabilities follow:

$$P = \begin{bmatrix} 0 & 0 & 0 & 0.5 & 0.25 & 0.25 & 0 \\ 0 & 0.25 & 0.25 & 0.5 & 0 & 0 & 0 \\ 2/3 & 1/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/3 & 2/3 \end{bmatrix} \quad (1.6)$$

1.3.2.2 Online HMM Learning

An online EM learning is first performed at each iteration to calculate and update HMM parameters that are re-estimated upon observing each new sub-sequence.

Particles positions and velocities vary over iterations, impacting also the evolutionary factor. Then, the classification environment for HMM changes during operations. Online learning of new data sequences allows adapting HMM parameters as new data become available as shown if Fig. 1.2, where $(o_i)_{i \in [0,n]}$ are data observations and $(\lambda_i)_{i \in [0,n]}$ parameters values update.

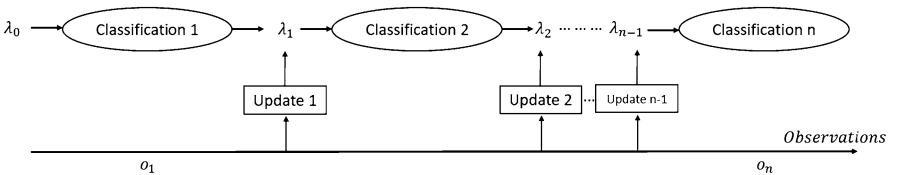


Fig. 1.2 HMM online classifications

At each iteration t , a new classifier r_t is performed with new updated parameters. We choose online learning EM algorithm [3] instead of Bach learning (classical

Baum-Welch algorithm [2], because this last one needs to run on all observation sequence which this is not our case.

The online Expectation-Maximization algorithm used for HMM parameters learning can be summarized as follows:

Algorithm 1 Expectations-maximization algorithm

Observation sequence $O = (o_1 o_2 \dots o_N)$

Initialization: initial parameters set λ_0 ;

for $i = 1$ to N_{step} : **do**

E-step: find conditionally optimal hidden trace S^i :

$S^i = \arg \max_s P(O|S, \lambda_{i-1})$;

 Compute likelihood

$L^{i-1}(\lambda) = P(O|S^i, \lambda_{i-1})$;

if ($i \leq N_{step}$ & likelihood not yet converged) **then**

M-step - find conditionally optimal parameter set λ :

$\lambda_i = \arg \max_s P(O|S, \lambda)$

end if

end for

Results Estimated parameters $\lambda_{N_{step}}$

1.3.2.3 HMM Classification

The Viterbi Algorithm is used with online parameters setting to find the most probable sequence of hidden states with a given sequence of observed states. The Viterbi algorithm does not simply accept the most likely state at a particular time instant, but also takes a decision based on the whole observation sequence. The algorithm will find the max Q (state sequence $Q = q_1 q_2 \dots q_T$) for a given observation sequence by the means of induction. An array $\psi_i(j)$ is used to store the highest probability paths [22].

HMM parameters, HMM classification is done by the Viterbi algorithm as defined in the next algorithm:

1.3.2.4 Our Algorithm

Therefore, we delegate choosing states of APSO iterations to online HMM classification, transitions between states is represented by online probabilities transitions. At each iteration transition and observation probabilities are updated according to the online EM algorithm. The Viterbi Algorithm is then used for state classification of APSO iteration. Then, we update positions and velocities according to the classified state. The complete hybrid APSO with HMM is depicted below (Algorithm 3).

Algorithm 2 Viterbi algorithm

Data: Observations of length T , state-graph of length N
Initialization: Observations of length T , state-graph of length N
 Create a path probability matrix viterbi $[N + 2, T]$
 Create a path backpointer matrix backpointer $[N + 2, T]$
for $s = 1$ to N **do**
 $forward[s, 1] \rightarrow a_{o,s} x b_s(o_1)$
 $backpointer[s, 1] \rightarrow 0$
end for
for time step t from 2 to T **do**
 for state s from 1 to N **do**
 $viterbi[s, t] \rightarrow \max_{s'=1}^N viterbi[s', t-1] x a_{s',s} x b_s(o_t)$
 $backpointer[s, t] \rightarrow \arg \max_{s'=1}^N viterbi[s', t-1] x a_{s',s}$
 end for
end for
 $t \rightarrow t + 1$
Result: Best-path of states: the classified current state

Algorithm 3 HMM-APSO algorithm

Data: The objective function
Initialization: positions, velocities of particles, accelerations factors and HMM parameters
 Set t value to 0
while (number of iterations $t \leq t_{max}$ not met) **do**
 Update HMM parameters by online EM process (algorithm 1)
 Classification of PSO state by HMM classifier (algorithm 2)
 Update c_1 , c_2 and w values according to the corresponding state
 for $i = 1$ to number of particles **do**
 compute f
 Update velocities and positions according to Eqs. (2) and (3)
 if $f \leq f_{best}$ **then**
 $f_{best} \rightarrow f$
 $p_{best} \rightarrow X$
 end if
 if $f(p_{best}) \leq f(g_{best})$ **then**
 $f(g_{best}) \rightarrow f_{best}$
 $g_{best} \rightarrow X_{best}$
 end if
 if state = convergence **then**
 Elastic learning
 end if
 end for
 $t \rightarrow t + 1$
end while
Result: The solution based on the best particle in the population and corresponding fitness value

1.4 Experiment

In this part, tests and validations of the proposed hybrid approach HMM-APSO are performed. Experimentations are done with several benchmark functions and compared with other PSO's variants of literature.

1.4.1 Parameters Setting

For each of benchmark functions shown in Table 1.1, we perform ten executions, and compare for each function the best and the average value.

Table 1.1 Description of Benchmark functions

Test functions	Name	Type
$f_1 = \sum_{i=1}^D [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	Rosenbrock	Unimodal
$f_2 = \sum_{i=1}^D (x_i + 0.5)^2$	Step	Unimodal
$f_3 = \sum_{i=1}^D x_i^2$	Sphere	Unimodal
$f_4 = 10^6 x_1^2 + \sum_{i=2}^D x_i^2$	Tablet	Unimodal
$f_5 = \sum_{i=1}^D (\sum_{j=1}^D x_j)^2$	Quadric	Unimodal
$f_6 = \sum_{i=1}^D [x_i^2 - 10 \cos(2\pi x_i) + 10]$	Rastrigrin	Multimodal
$f_7 = -20 \exp(-0.2 \sqrt{\frac{1}{D} x_i^2})$	Ackley	Multimodal
$f_8 = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \Pi \cos(x_i / \sqrt{i}) + 1$	Griewang	Multimodal
$f_9 = \sum_{i=1}^D x_i \sin(\sqrt{ x_i })$	Schwefel	Multimodal
$f_{10} = -\frac{(1 + \cos(12\sqrt{x_1^2 + x_2^2}))}{\frac{1}{2}(x_1^2 + x_2^2) + 2}$	Drop wave	Multimodal

Table 1.2 shows ten improved variants of PSO found in the literature. Tests are executed with the same value of acceleration factors and inertia weight coefficients which are:

$$c_1 = c_2 = 2, \omega = 0.9$$

The used swarm population size is 30 with a dimension of 30. Each run contains 1000 generations of the optimization process. Performance is qualified following two main measured observations: comparison on the solution accuracy and comparison on the convergence speed.

Table 1.2 Compared variants of PSO

Algorithm	Name	Specific parameters	Reference
YPSO	PSO with compressibility factor	–	[17]
SELPSO	Natural selection based PSO	–	[11]
SecVibratPSO	Order oscillating PSO	–	[12]
SecPSO	Swarm-core evolutionary PSO	–	[28]
SAPSO	Self-adaptive PSO	$\omega_{min} = 0.01$	[11]
RandWPSO	Random inertia weight PSO	$mean_{max} = 1, mean_{min} = 0$	[29]
LinWPSO	Linear decreasing weights PSO	$\omega_{min} = 0.0001, \omega_{max} = 0.1$	[29]
CLSPSO	Cooperative line search PSO	–	[28]
AsyLnCPSO	Asynchronous PSO	$c_{1min} = c_{2min} = 0.01,$ $c_{1max} = c_{2max} = 2$	[11]
SimuAPSO	PSO with simulated annealing	$\lambda = 0.0001,$	[28]

1.4.2 Comparison on the Solution Accuracy

To examine our HMM-APSO approach, we compared results obtained for the benchmark test functions with APSO. For every benchmark function, executions are performed for all variants of PSO. Mean and best values are calculated to evaluate the solution accuracy among other PSO variants from literature (Table 1.2). We depict the obtained results in Table 1.3.

Table 1.3 Results comparisons with the variants of PSO

Functions	APSO	PSO	SimuA-PSO	Sec-PSO	RandWPSO	YPSO	SelPSO	SecVibratPSO	SAPSO	LinWPSO	AsyLnCPSO	HMM-APSO
f_1	Best 49	13566	115719	4689	9843	1420	16382	275	5095	7067	3765	44
	Mean 150	24618	288102	10930	33384	2726	27998	32132	14850	20280	11045	171
f_2	Best 0	5.1e-09	5.8e-06	1.9e-31	2.1e-12	0	8.6e-10	7.3e-10	0	0	0	0
	Mean 0	6.5e-05	0.04	9.8e-12	3.6e-05	0	1.8e-05	0.03	0	0	3.1e-30	0
f_3	Best 0.01	26.55	93.19	20.58	37.7	6.77	36.01	0.8	18.20	17.66	22.89	4.6e-3
	Mean 0.05	50.15	188.56	38.79	64.09	13.77	63.59	71.05	31.96	40.53	34.58	0.04
f_4	Best 0.02	94.97	251.36	71.4	110.48	23.72	30.47	17.42	37.21	51.24	21.73	0.02
	Mean 0.05	135.82	396.21	110.48	246.78	42.31	77.51	199.04	84.79	95.74	44.32	0.07
f_5	Best 16435	629e+5	587e+6	421e+5	102e+6	127e+4	839e+5	107e+6	459e+5	165e+5	131e+5	7644
	Mean 67851	196e+6	210e+7	978e+5	434e+6	843e+4	210e+6	562e+6	166e+6	155e+6	384e+5	49205
f_6	Best 8.24	266.20	358.22	208.54	293.88	142.44	285.16	221.93	165.66	170.64	193.83	4.31
	Mean 16.14	307.24	462.02	262.89	322.35	175.84	315.60	344.76	273.31	261.77	291.54	12.41
f_7	Best 0.03	4.79	6.9436	4.8963	5.403	3.1785	5.665	1.2753	3.8279	4.7261	5.8372	0.03
	Mean 0.31	5.61	8.6336	5.2716	6.5613	4.2219	6.1951	4.4528	5.2531	5.6829	6.8189	0.33
f_8	Best 7.50e-5	0.17	0.52	0.15	0.25	0.05	0.27	0.05	0.13	0.14	0.08	1.1e-5
	Mean 0.01	0.39	0.87	0.25	0.45	0.12	0.41	0.42	0.25	0.31	0.23	0.07
f_9	Best -118.3	-3e+28	-7.2+4	-4+158	-	-3+34	-1e+3	-	-1+231	-1e+22	-3e+47	-118.3
	Mean -118.3	-3e+28	-1e+47	-9e+15	-	-3e+33	-	-	-1e+23	-1e+21	-3e+46	-118.3
f_{10}	Best -1	-1	-0.92	-1	-0.94	-1	-0.99	-0.93	-1	-1	-1	-1
	Mean -1	-0.95	-0.74	-0.96	-0.93	-0.98	-0.95	-0.82	-0.97	-0.96	-0.98	-1

Results obtained from the mean of all executions shows that for these benchmark functions ($f_1, f_2, f_5, f_6, f_7, f_8, f_9, f_{10}$) HMM-APSO gives almost the best results among others used PSO variants. In general HMM-APSO gives good accuracy.

1.4.3 Comparison on the Convergence Speed

In this section, we display in the following figures the obtained values at each iteration:

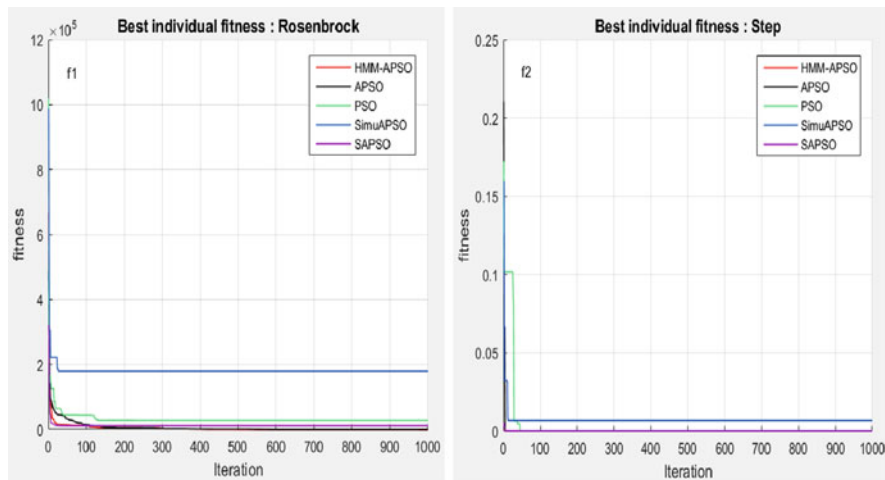


Fig. 1.3 Comparison on Rosenbrock and Step functions

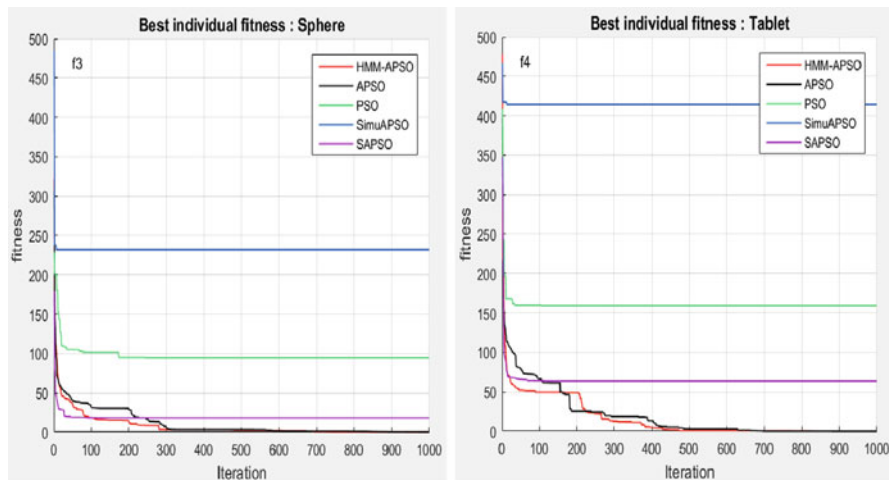


Fig. 1.4 Comparison on sphere and tablet functions

HMM-APSO gives frequently the best solution. When we interest on the convergence rate, we can notice from Figs. 1.3, 1.4, 1.5, 1.6, and 1.7, that HMM-APSO has faster convergence rate than other PSO.

We can conclude from experimentation that HMM-APSO can improve APSO when comparing to others PSO with a number of benchmark functions. However, those results improve the online HMM classification methodology. HMM coupled

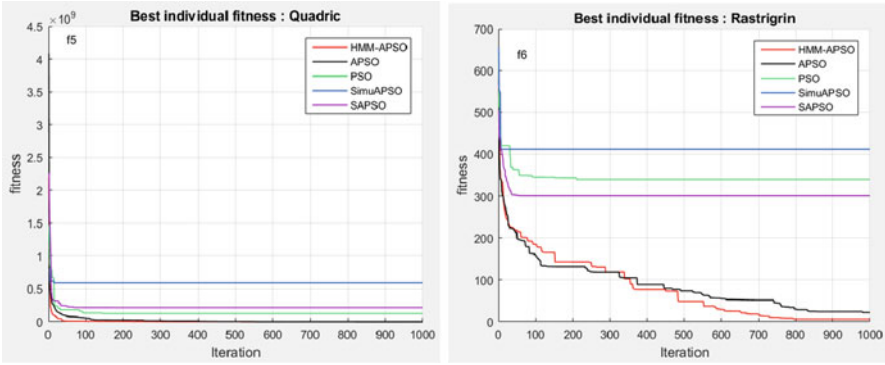


Fig. 1.5 Comparison on quadric and Rastrigrin functions

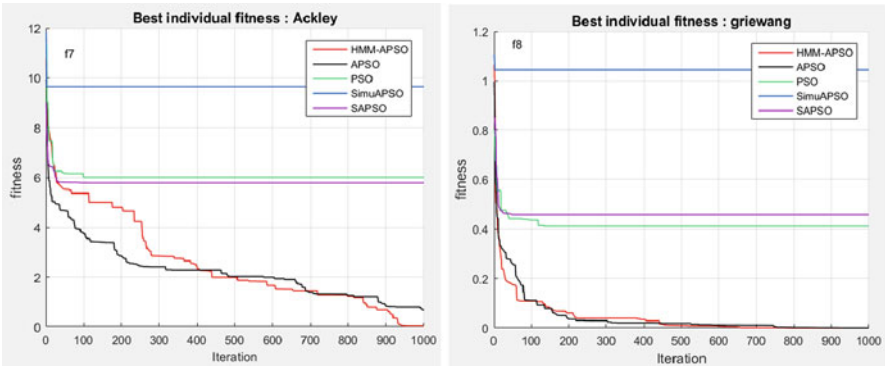


Fig. 1.6 Comparison on Ackley and Griewang functions

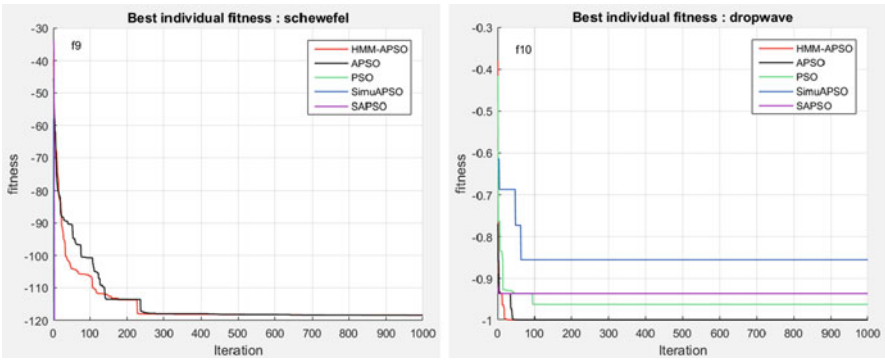


Fig. 1.7 Comparison on Schwefel and drop wave functions

with unsupervised learning EM, gives more state adaptation to the APSO algorithm. So, HMM-APSO outperforms significantly the original APSO algorithm.

1.5 Conclusion

In this paper, we have integrated online HMM classifier inside the APSO. This looks advantageous from the view that HMM is a robust stochastic classification tool. The HMM can include stochastic information on transitions between states and also observations to give the most likely state for the APSO. The online EM process gives more adaptive capacity to environment change during iteration which gives more quality to the HMM classifier. We benefit from this feature of HMM classification to integrate it inside APSO. On the other hand, we build an unsupervised state control of APSO iteration will be provided by online HMM. This approach gives better results than the majority of the state of art of PSO improvement in terms of both solution accuracy and convergence speed. Future research should attempt to provide more control to HMM inside the PSO algorithm in order to describe more the stochastic PSO behavior through iterations. That is, by using Online learning HMM in the convergence state instead of ELS procedure, it may further enhance APSO more performance, or even constitute a new all based HMM learning for PSO, not only for classification process.

References

1. G. Ardizzon, G. Cavazzini, G. Pavesi, Adaptive acceleration coefficients for a new search diversification strategy in particle swarm optimization algorithms. *Inf. Sci.* **299**, 337–378 (2015)
2. L.E. Baum, T. Petrie, G. Soules, N. Weiss, A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Stat.* **41**, 164–171 (1970)
3. O. Cappé, Online Em algorithm for hidden Markov models. *J. Comput. Graph. Stat.* **20**(3), 728–748 (2011)
4. C. DeBao, Z. ChunXia, Particle swarm optimization with adaptive population size and its application. *Appl. Soft Comput.* **9**, 39–48 (2009)
5. N. Di Mauro, F. Esposito, S. Ferilli, T.M.A. Basile, Avoiding order effects in incremental learning, in *AI*IA 2005: Advances in Artificial Intelligence*, ed. by S. Bandini, S. Manzoni. Lecture Notes in Computer Science, vol. 3673 (Springer, Berlin, 2005), pp. 110–121
6. J. Ding, J. Liu, K.R. Chowdhury, W. Zhang, Q. Hu, J. Lei, A particle swarm optimization using local stochastic search and enhancing diversity for continuous optimization. *Neurocomputing* **137**, 261–267 (2014)
7. M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Evolving cognitive and social experience in particle swarm optimization through differential evolution: a hybrid approach. *Inf. Sci.* **216**, 50–92 (2012)
8. A.B. Hashemi, M.R. Meybodiv, A note on the learning automata based algorithms for adaptive parameter selection in PSO. *Appl. Soft Comput.* **11**(1), 689–705 (2011)
9. H. Huang, H. Qin, Z. Hao, A. Lim, Example-based learning particle swarm optimization for continuous optimization. *Inf. Sci.* **128**, 125–138 (2012)
10. S.S. Jadon, H. Sharma, J.C. Bansal, R. Tiwari, Self adaptive acceleration factor in particle swarm optimization, in *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications*, vol. 1 (Springer India, New Delhi, 2013), pp. 325–340

11. W. Jiang, Y. Zhang, R. Wang, Comparative study on several PSO algorithms, in *The 26th Chinese Control and Decision Conference (2014 CCDC)*, May 2014, pp. 1117–1119
12. H. Jianxiu, Z. Jianchao, A two-order particle swarm optimization model [J]. *J. Comput. Res. Dev.* **11**, 004 (2007)
13. S.M. Kamalapur, V.H. Patil, Impact of acceleration coefficient strategies with random neighborhood topology in particle swarm optimization. *Int. J. Emerg. Technol. Comput. Appl. Sci.* **3**(1), 37–42 (2012)
14. J. Kennedy, R.C. Eberhart, Particle swarm optimization, in *Proceedings of IEEE International Conference Neural Networks* (IEEE, New York, 1995), pp. 1942–1948
15. A. Khare, S. Rangnekar, A review of particle swarm optimization and its applications in solar photovoltaic system. *Appl. Soft Comput.* **13**, 2997–3006 (2013)
16. X. Liang, W. Li, Y. Zhang, M.C. Zhou, An adaptive particle swarm optimization method based on clustering. *Soft Comput.* **19**(2), 431–448 (2015)
17. L.-l. Liu, X.-B. Gao, An adaptive simulation of bacterial foraging algorithm. *Basic Sci. J. Text. Univ.* **4**, 022 (2012)
18. F. Pan, X.-T. Li, Q. Zhou, W.-X. Li, Q. Gao. Analysis of standard particle swarm optimization algorithm based on Markov chain. *Acta Automat. Sin.* **39**(4), 381–389 (2013)
19. R.E. Perez, K. Behdinan, Particle swarm approach for structural design optimization. *Comput. Struct.* **85**, 1579–88 (2007)
20. S. Phon-Amnuaisuk, Estimating HMM parameters using particle swarm optimisation, in *Applications of Evolutionary Computing*, ed. by M. Giacobini, A. Brabazon, S. Cagnoni, G.A. Di Caro, A. Ekart, A.I. Esparcia-Alcazar, M. Farooq, A. Fink, P. Machado. *Lecture Notes in Computer Science*, vol. 5484 (Springer, Berlin, 2009), pp. 625–634
21. R. Polikar, L. Upda, S.S. Upda, V. Honavar, Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **31**(4), 497–508 (2001)
22. L. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* **77**(2), 257–286 (1989)
23. Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in *Proceeding of the IEEE World Congress of Computational Intelligence*, 1997, pp. 69–73
24. P. Subbaraj, R. Rengaraj, S. Salivahanan, T.R. Senthilkumar, Parallel particle swarm optimization with modified stochastic acceleration factors for solving large scale economic dispatch problem. *Int. J. Electr. Power Energy Syst.* **32**(9), 1014–1023 (2010)
25. S. Sun, H. Liu, Particle swarm algorithm: convergence and applications, in *Swarm Intelligence and Bio-Inspired Computation*, ed. by X.-S. Yang, Z. Cui, R. Xiao, A.H. Gandomi, M. Karamanoglu (Elsevier, Amsterdam, 2013), pp. 137–168
26. Y. Tang, Z. Wang, J.A. Fang, Feedback learning particle swarm optimization. *Appl. Soft Comput.* **11**, 4713–4725 (2011)
27. F. van den Bergh, A. Engelbrecht, A cooperative approach to particle swarm optimization. *IEEE Trans. Evol. Comput.* **8**(3), 225–239 (2004)
28. S. Wang, M. Chen, D. Huang, X. Guo, C. Wang, Dream effected particle swarm optimization algorithm. *J. Inf. Comput. Sci.* **11**(15), 5631–5640 (2014)
29. Z Wu, Optimization of distribution route selection based on particle swarm algorithm. *Int. J. Simul. Model.* **13**(2), 230–242 (2014)
30. F. Yang, C. Zhang, An effective hybrid optimization algorithm for HMM, in *Fourth International Conference on Natural Computation, 2008. ICNC '08*, Oct 2008, vol. 4, pp. 80–84
31. X. Yu, X. Zhang, Enhanced comprehensive learning particle swarm optimization. *Appl. Math. Comput.* **242**, 265–276 (2014)
32. Z.-H. Zhan, J. Zhang, Y. Li, H.S.-H. Chung, Adaptive particle swarm optimization. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **39**(6), 1362–1381 (2009)
33. Z. Zhan, J. Zhang, Y. Li, Y.H. Shi, Orthogonal learning particle swarm optimization. *IEEE Trans. Evol. Comput.* **15**, 832–847 (2011)
34. L. Zhang, Y. Tang, C. Hua, X. Guan, A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques. *Appl. Soft Comput.* **28**, 138–149 (2015)

Chapter 2

Possibilistic Framework for Multi-Objective Optimization Under Uncertainty

Oumayma Bahri, Nahla Ben Amor, and El-Ghazali Talbi

Abstract Optimization under uncertainty is an important line of research having today many successful real applications in different areas. Despite its importance, few works on multi-objective optimization under uncertainty exist today. In our study, we address combinatorial multi-objective problem under uncertainty using the possibilistic framework. To this end, we firstly propose new Pareto relations for ranking the generated uncertain solutions in both mono-objective and multi-objective cases. Secondly, we suggest an extension of two well-known Pareto-base evolutionary algorithms namely, SPEA2 and NSGAI. Finally, the extended algorithms are applied to solve a multi-objective Vehicle Routing Problem (VRP) with uncertain demands.

Keywords Multi-objective optimization • Uncertainty • Possibility theory • Evolutionary algorithms • Vehicle routing problem

2.1 Introduction

Most real-world decision problems are multi-objective in nature as they require the simultaneous optimization of multiple and usually conflicting objectives. These multi-objective problems are a very important and widely discussed research topic. Yet, despite the massive number of existing resolution methods and techniques for multi-objective optimization, there still many open questions in this area. In fact, there is no consideration of uncertainty in the classical multi-objective concepts and techniques, which makes their application to real-life optimization problems impossible.

O. Bahri
LARODEC and INRIA Lab., University Lille 1, Lille, France

N. Ben Amor
LARODEC Laboratory, ISG Tunis, Le Bardo, Tunisia

E.-G. Talbi (✉)
INRIA Laboratory, CRISTAL/CNRS, University Lille 1, Villeneuve d'Ascq, France
e-mail: el-ghazali.talbi@univ-lille1.fr

Moreover, uncertainty characterizes almost all practical applications, in which the big amount of data provides certainly some unavoidable imperfections. This imperfection might result from using unreliable information sources caused by inputting data incorrectly, faulty reading instruments or bad analysis of some training data. It may also be the result of poor decision-maker opinions due to any lack of its background knowledge or even due to the difficulty of giving a perfect qualification for some costly situations. The classical way to deal with uncertainty is the probabilistic reasoning, originated from the middle of the seventeenth century [19]. However, probability theory was considered for a long time as a very good quantitative tool for uncertainty treatment, but as good as it is, this theory is only appropriate when all numerical data are available, which is not always the case. Indeed, there are some situations such as the case of total ignorance, which are not well handled and which can make the probabilistic reasoning unsound [26]. Therefore, a panoply of non-classical theories of uncertainty have recently emerged such as fuzzy sets theory [33], possibility theory [34] and evidence theory [25]. Among the aforementioned theories of uncertainty, our interest will focus on possibility theory which offers a natural and simple model to handle uncertain data and presents an appropriate framework for experts to express their partial beliefs numerically or qualitatively. Nevertheless, while the field of optimization under uncertainty has gained considerable attention during several years in the mono-objective context, only few studies have been focused on treating uncertain optimization problems within a multi-objective setting. This chapter addresses the multi-objective optimization problems under uncertainty in the possibilistic setting [23].

The remainder of the chapter is organized as follows. Section 2.2 recalls the main concepts of deterministic multi-objective optimization. Section 2.3 gives an overview of existing approaches for multi-objective optimization under uncertainty. Section 2.4 presents in detail our proposed possibilistic framework after briefly recalling the basics of possibility theory. Finally, Sect. 2.5 describes an illustrative example on a multi-objective vehicle routing problem with uncertain demands and summarizes the obtained results.

2.2 Background on Deterministic Multi-Objective Optimization

Deterministic multi-objective optimization is the process of optimizing systematically and simultaneously two or more conflicting objectives subject to certain constraints. In contrast to mono-objective optimization, a multi-objective optimization problem does not restrict to find a unique global solution but it aims to find the most preferred exact solutions among the best ones.

Formally, a basic multi-objective optimization problem (MOP), defined in the sense of minimization of all the objectives, consists of solving a mathematical program of the form:

$$MOP = \begin{cases} \text{Min } F(x) = (f_1(x), f_2(x), \dots, f_n(x)) \\ \text{s.t. } x \in S \end{cases} \quad (2.1)$$

where n ($n \geq 2$) is the number of objectives and $x = \{x_1, \dots, x_k\}$ is the set of decision variables from the decision space S , which represents the set of feasible solutions associated with equality and inequality constraints. $F(x)$ is the vector of independent objectives to be minimized. This vector F can be defined as a cost function in the objective space by assigning an objective vector \vec{y} which represents the quality of the solution (or fitness).

$$F : X \rightarrow Y \subseteq \mathbb{R}^n, \quad F(x) = \vec{y} = \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix} \quad (2.2)$$

In order to identify better solutions of a given MOP, other concepts of optimality should be applied such as *Pareto dominance*, *Pareto optimality*, *Pareto optimal set* and *Pareto front*. Without loss of generality, we assume that the sense of minimization of all the objectives is considered in the following concepts definition:

An objective vector $\mathbf{x} = (x_1, \dots, x_n)$ is said to *Pareto dominate* another objective vector $\mathbf{y} = (y_1, \dots, y_n)$ (denoted by $\mathbf{x} \prec_p \mathbf{y}$) if and only if no component of \mathbf{y} is smaller than the corresponding component of \mathbf{x} and at least one component of \mathbf{x} is strictly smaller:

$$\forall i \in 1, \dots, n : x_i \leq y_i \wedge \exists i \in 1, \dots, n : x_i < y_i. \quad (2.3)$$

For a minimization $MOP(F, S)$, a solution $x^* \in X$ is *Pareto optimal* (also known as efficient, non-dominated or non-inferior) if for every $x \in X$, $F(x)$ does not dominate $F(x^*)$, that is, $F(x) \not\prec_p F(x^*)$.

A *Pareto optimal set* P^* is defined as:

$$P^* = \{x \in X / \exists x' \in X, F(x') \not\prec_p F(x)\}. \quad (2.4)$$

The image of this *Pareto optimal set* P^* in the objective space is called *Pareto front* PF^* defined as:

$$PF^* = \{F(x), x \in P^*\}. \quad (2.5)$$

Yet, finding the true *Pareto front* of a general MOP is NP-hard. Thus, the main goal of multi-objective optimization is to identify a good approximation of the *Pareto front*, from which the decision maker can select an optimal solution based on the current situation. The approximated front should satisfy two properties: (1) convergence or closeness to the exact Pareto front and (2) uniform diversity of the obtained solutions around the Pareto front. Figure 2.1 illustrates an example of approximated front having a very good spread of solutions (uniform diversity) but a bad convergence, since the solutions are far from the true Pareto front.

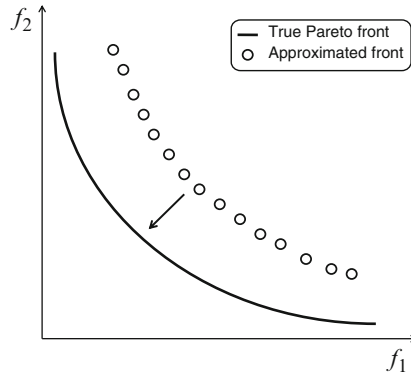


Fig. 2.1 Example of Pareto front with uniform diversity and bad convergence

There are several deterministic optimization methods to deal with multi-objective combinatorial problems, such as the metaheuristics, which mark a great revolution in the field of optimization. A review of various metaheuristics can be found in [29]. Among the well-know metaheuristics, evolutionary algorithms seem particularly suitable for both theoretical and practical MOPs, since they have the ability to search partially ordered spaces for several alternative trade-offs [6, 5, 7]. Some of the most popular multi-objective evolutionary algorithms (MOEAs) are: Multi-Objective Genetic Algorithm (MOGA) [11], Niche-Pareto Genetic Algorithm (NPGA) [14], Pareto-Archived Evolutionary Strategy (PAES) [18], Strength Pareto Evolutionary Algorithms (SPEA, SPEA2) [35, 36] and Non-dominated Sorting Genetic Algorithms (NSGA, NSGAI) [8, 9]. Such algorithms are based on three main components namely, Fitness assignment, Diversity preserving and Elitism.

Fitness Assignment

Fitness Assignment allows to guide the search algorithm toward Pareto optimal solutions for a better convergence. The fitness assignment procedure assigns to each objective vector, a scalar-valued fitness that measures the quality of solution. According to the fitness assignment strategy, four different categories can be identified:

- Pareto-based assignment: based on the concept of dominance and Pareto optimality to guide the search process. The objective vectors are scalarized using the dominance relation.
- Scalar-based assignment: based on the MOP transformation into a mono-objective problem by using for example aggregation methods and weighted metrics.
- Criterion-based assignment: based on the separate handling of various non commensurable objectives by performing a sequential search according to a given preference order of objectives or by handling the objectives in parallel.
- Indicator-based assignment: based on the use of performance quality indicators to drive the search toward the Pareto front.

Diversity Preserving

Diversity Preserving used to generate a diverse set of Pareto solutions. According to the strategy of density estimation, three categories can be distinguished:

- Distance-based density assessment: based on the distance between individuals in the feature space. Examples of techniques are, Niche sharing, Clustering, K th nearest neighbor and Crowding.
- Grid-based density assessment: based on the way in which a number of individuals residing within predetermined cells are located. Histogram method is an example.
- Distribution-based density assessment: based on the probability density of individuals using for example probability density estimation functions.

Elitism

Elitism consists in archiving the best solutions found (e.g, Pareto optimal solutions) in order to prevent the loss of good solutions during the search process. Archiving process can be done using an archive (elite population) or an external population and its strategy of update usually relies on size, convergence and diversity criteria. Depending on the manner in which the archiving process is performed, MOEAs can be classified into two categories, namely non-elitist and elitist MOEAs. Moreover, almost all MOEAs follow the same basic steps in the search process [12], as outlined in the following pseudo code:

Generic MOEA Framework

```

Initialize random population P
While (Stopping condition is not satisfied)
    Fitness evaluation of solutions in P;
    Environmental selection of "good" solutions;
    Diversity preserving of candidate solutions;
    Update and store elite solutions into an external population or archive;
    Mating selection to create the mating pool for variation;
    Variation by applying crossover and mutation operators;
End While

```

An MOEA begins its search with a population of solutions usually generated at random. Thereafter, an iterative optimization process takes place by the use of six search operators: evaluation of the population individuals, environmental selection to choose better solutions based on their fitness, diversity preservation of candidate solutions, updating and archiving the solutions into an external population or archive, mating selection operator in which solutions are picked from the updated population to fill an intermediate mating pool and finally variation operator to generate new solutions. The process stops when one or more pre-specified stopping conditions are met.

All the above concepts and techniques of deterministic multi-objective optimization are widely used and applied successfully to several combinatorial decision problems in many interesting areas, but their application to real-life decision making situations often faces some difficulties. Yet, most of real-world optimization problems are naturally subject to various types of uncertainties caused by many sources such as missing information, forecasting, data approximation or noise in measurements. These uncertainties are very difficult to avoid in practical applications and so should be taken into account within the optimization process. Therefore, a variety of methodologies and approaches for handling optimization problems under uncertainty have been proposed in the last years. Unfortunately, almost all of them have been devoted to solve such problems in the mono-objective context, while only few studies have been performed in the multi-objective setting. A review of some existing approaches for uncertain multi-objective optimization will be summarized in the next section.

2.3 Existing Approaches for Uncertain Multi-Objective Optimization

Uncertain multi-objective optimization has gained more and more attention in recent years [17], since it closely reflects the reality of many real-world problems. Such problems, known as multi-objective problems under uncertainty, are naturally characterized by the necessity of optimizing simultaneously several objectives subject to a set of constraints and while considering that some input data are ill-known and without knowing what their full effects will be. In these problems, the set of objectives and/or constraints to be satisfied can be affected by the uncertainty of input data or uncontrollable problem parameters. Hence, the aim of optimization in this case will be to find solutions of a multi-objective problem that are not only feasible and optimal but also their objectives and/or constraints are allowed to have some acceptable (or minimal) uncertainties. These uncertainties can take different forms in terms of distribution, bounds, and central tendency.

Yet, considering the uncertainty in the objective functions seems to be very applicable but highly critical, since the propagation of input uncertainties to the objectives may have a major impact on the whole optimization process and consequently on the problem solutions. In most of the existing approaches for dealing with multi-objective problems under uncertainty, the objective functions to be optimized are transformed into different forms in order to simplify their resolution by eliminating one of the two basic characteristics of such problems: multi-objectivity and uncertainty propagation. In fact, some of these approaches have been often limited to simply reduce the problem to mono-objective context by considering the set of objectives as if there's only one, using for example an aggregation function (a weighted sum) of all the objectives [13] or preferring only one objective to be optimized (based on a preference indicator) and fixing the remaining objectives as constraints [24]. The considered single objective is then optimized using appropriate mono-objective methods for uncertainty treatment.

Some other approaches have been focused on treating the problem as multi-objective but with ignorance of uncertainty propagation to the objective functions by converting them into deterministic functions using statistical properties. For example, in [30], expectation values are used to approximate the observed interval-valued objectives and so the goal became to optimize the expected values of these objectives. In [2], the average value per objective is firstly computed and then a ranking method based on the average values of objectives is proposed. Similarly, [9] suggested to consider the mean value for each objective vectors and then to apply classical deterministic multi-objective optimizers. Nevertheless, the uncertainty of objective values must not be ignored during the optimization process, because if the input data or parameters are highly uncertain, how can the optimizer simply state that the uncertainty of outputs is completely certain? It may be feasible only for simplicity or other practical reasons as long as the algorithm performance will not be affected.

To this end, some distinct approaches have been suggested to handle the problem as-is without erasing any of its multi-objective or uncertain characteristics by introducing a particular multi-objective optimizer for this purpose. Indeed, [21, 22, 1] proposed to display uncertainty in objective functions through intervals of belief functions and then introduced an extensions of Pareto dominance for ranking the generated interval-valued objectives. Hughes [15, 16] suggested to express uncertainty in the objectives via special types of probability distributions and then independently proposed a stochastic extension of Pareto dominance. Our interest in this chapter will focus on handling multi-objective problems under uncertainty in the possibilistic setting while considering the uncertainty propagation to the set of objectives to be optimized.

2.4 Proposed Possibilistic Framework for Multi-Objective Problems Under Uncertainty

This section provides firstly a brief background on possibility theory and then presents in detail the proposed possibilistic framework for solving multi-objective problems with uncertain data. The framework is composed of three main stages: Adaptation of possibilistic setting, New Pareto optimality and Extension of some optimization algorithms to our uncertain context.

2.4.1 Basics on Possibility Theory

Possibility theory, issued from Fuzzy Sets theory, was introduced by Zadeh [34] and further developed by Dubois and Prade [10]. This theory offers a flexible tool for representing uncertain information such as expressed by humans. Its basic building block is the notion of possibility distribution, denoted by π and defined as the following:

Let $V = \{X_1, \dots, X_n\}$ be a set of state variables whose values are ill-known. We denote by x_i any instance of X_i and by D_{X_i} the domain associated with X_i . $\Omega = D_{X_1} \times \dots \times D_{X_n}$ denotes the universe of discourse, which is the cartesian product of all variable domains V . Vectors $\omega \in \Omega$ are often called realizations or simply “states” (of the world). The agent’s knowledge about the value of the x_i ’s can be encoded by a possibility distribution π that corresponding to a mapping from the universe of discourse Ω to the scale $[0, 1]$, i.e. $\pi : \Omega \rightarrow [0, 1]$; $\pi(\omega) = 1$ means that the realization ω is totally possible and $\pi(\omega) = 0$ means that ω is an impossible state. It is generally assumed that there exist at least one state ω which is totally possible— π is said then to be *normalized*. Extreme cases of knowledge are presented by:

- *complete knowledge* i.e. $\exists \omega_0 \in \Omega, \pi(\omega_0) = 1$ and $\forall \omega \neq \omega_0, \pi(\omega) = 0$.
- *total ignorance* i.e. $\forall \omega \in \Omega, \pi(\omega) = 1$ (all values in Ω are possible).

From π , one can describe the uncertainty about the occurrence of an event $A \subseteq \Omega$ via two dual measures: the possibility $\Pi(A)$ and the necessity $N(A)$ expressed by:

$$\Pi(A) = \sup_{\omega \in A} \pi(\omega). \quad (2.6)$$

$$N(A) = 1 - \Pi(\neg A) = 1 - \sup_{\omega \notin A} \pi(\omega) \quad (2.7)$$

Measure $\Pi(A)$ corresponds to the possibility degree (i.e. the plausibility) of A and it evaluates to what extent A is consistent (i.e. not contradictory) with the knowledge represented by π . Yet, the expression “*it is possible that A is true*” does not entail anything about the possibility nor the impossibility of A . Thus, the description of uncertainty about the occurrence of A needs its dual measure $N(A)$ which corresponds to the extent to which A is impossible and it evaluates at which level A is certainly implied by the π (the certainty degree of A). Main properties of these two dual measures are summarized in Table 2.1.

Table 2.1 Possibility measure Π and necessity measure N

$\Pi(A) = 1$ and $\Pi(\bar{A}) = 0$	$N(A) = 1$ and $N(\bar{A}) = 0$	A is certainly true
$\Pi(A) = 1$ and $\Pi(\bar{A}) \in]0, 1[$	$N(A) \in]0, 1[$ and $N(\bar{A}) = 0$	A is somewhat certain
$\Pi(A) = 1$ and $\Pi(\bar{A}) = 1$	$N(A) = 0$ and $N(\bar{A}) = 0$	Total ignorance

The particularity of the possibilistic scale is that it can be interpreted in two manners: in an ordinal manner, i.e. when the possibility degrees reflect only an ordering between the possible values and in a numerical manner, i.e. when the handled values make sense in the ranking scale.

Technically, a possibility distribution is a normal fuzzy set (at least one membership grade equals 1). Indeed, all fuzzy numbers can be interpreted as specific possibility distributions. More precisely, given a variable X whose values are restricted by a fuzzy set F characterized by its membership function μ_F , so that π_X is taken as equal to the membership function $\mu_F(x)$. Thus, the possibility and necessity measures will be expressed in terms of supremum degrees of the μ_F , i.e.

$\Pi(X) = \sup_{x \in X} \mu_F(x)$ and $N(X) = 1 - \sup_{x \notin X} \mu_F(x)$. In this work, we are interested in a particular form of possibility distributions, namely those represented by triangular fuzzy numbers and commonly known as triangular possibility distributions. A triangular possibility distribution π_X is defined by a triplet $[\underline{x}, \hat{x}, \bar{x}]$, as shown in Fig. 2.2, where $[\underline{x}, \bar{x}]$ is the interval of possible values called its bounded support and \hat{x} denotes its kernel value (the most plausible value).

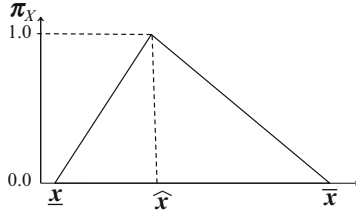


Fig. 2.2 Triangular possibility distribution

In the remaining, we use $X = [\underline{x}, \hat{x}, \bar{x}] \subseteq \mathbb{R}$ to denote the triangular fuzzy number X , meaning that X is represented by a triangular possibility distribution π_X . This representation is characterized by a membership function μ_X which assigns a value within $[0, 1]$ to each element in $x \in X$. Its mathematical definition is given by:

$$\mu_X(x) = \begin{cases} \frac{x-\underline{x}}{\hat{x}-\underline{x}}, & \underline{x} \leq x \leq \hat{x} \\ 1, & x = \hat{x} \\ \frac{\bar{x}-x}{\bar{x}-\hat{x}}, & \hat{x} \leq x \leq \bar{x} \\ 0, & \text{otherwise.} \end{cases} \quad (2.8)$$

However, in practical use of triangular fuzzy numbers, a ranking procedure needs to be applied for decision-making. In other words, one triangular fuzzy number needs to be evaluated and compared with the others in order to make a choice among them. Indeed, all possible topological relations between two triangular fuzzy numbers $A = [\underline{a}, \hat{a}, \bar{a}]$ and $B = [\underline{b}, \hat{b}, \bar{b}]$ may be covered by only four different situations, which are: Fuzzy disjoint, Fuzzy weak overlapping, Fuzzy overlapping and Fuzzy inclusion [20]. These situations, illustrated in Fig. 2.3 should be taken into account for ranking triangular fuzzy numbers.

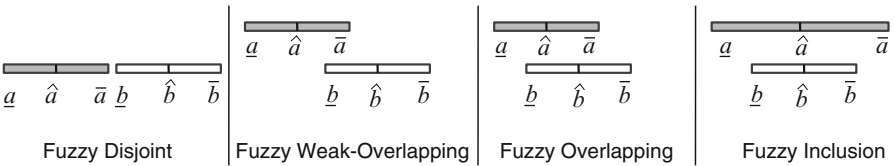


Fig. 2.3 Possible topological situations for two TFNs

2.4.2 Adaptation of Possibilistic Setting

In the following, we choose to express the uncertain data of multi-objective problems under uncertainty using triangular possibility distributions (i.e. triangular fuzzy numbers) as defined in the previous subsection. Then, as a multi-objective optimization problem under uncertainty involves the simultaneous satisfaction of several objectives respecting a set of constraints and while considering some input data uncertainties, we assume that the observed objectives and some constraints (especially those depends on uncertain variables) are affected by the used form of these uncertainties.

Thus, as in our case, uncertainty is represented by a triangular form, the uncertain constraints in such a problem may be disrupted by this fuzzy form and so will be fuzzy constraints. Yet, the satisfaction of such constraints cannot be directly predicted since it is difficult to estimate directly that a fuzzy constraint is fully satisfied or fully violated. At this level, we propose firstly to use the two measures of possibility theory Π and N in order to express the satisfaction of a given fuzzy constraint, as follows:

Let $X = [\underline{x}, \hat{x}, \bar{x}] \subseteq \mathbb{R}$ be a triangular fuzzy variable, let x be any instance of X , let v be a given fixed value and let $C = (X \leq v)$ be a fuzzy constraint that depends only on the value of X and whose membership function is $\mu(x)$, then we have the measures $\Pi(X \leq v)$ and $N(X \leq v) = 1 - \Pi(X > v)$ are equal to:

$$\Pi(\tilde{X} \leq v) = \sup \mu_{x \leq v}(x) = \begin{cases} 1 & \text{if } v > \hat{x} \\ \frac{v - \underline{x}}{\hat{x} - \underline{x}} & \text{if } \underline{x} \leq v \leq \hat{x} \\ 0 & \text{if } v < \underline{x}. \end{cases} \quad (2.9)$$

$$N(\tilde{X} \leq v) = 1 - \sup \mu_{x > v}(x) = \begin{cases} 1 & \text{if } v > \bar{x} \\ \frac{v - \hat{x}}{\bar{x} - \hat{x}} & \text{if } \hat{x} \leq v \leq \bar{x} \\ 0 & \text{if } v < \hat{x}. \end{cases} \quad (2.10)$$

These formulas will be used to express the degrees that a solution satisfies the fuzzy constraint.

Example 1 *As an example of constraint satisfaction expressed by the possibility and necessity measures, we have $Q = [\underline{q}, \hat{q}, \bar{q}] = [20, 45, 97]$ is a triangular fuzzy quantity of objects, $M = 50$ is the maximum size of a package and $C = (Q \leq M)$ is the fuzzy constraint which imposes that the total quantity of objects must be less than or equal to the package size. In this case, $\Pi(Q \leq M) = 1$ because $M = 50 > \hat{q} = 45$ and $N(Q \leq M) = \frac{M - \hat{q}}{\bar{q} - \hat{q}} = \frac{50 - 45}{97 - 45} = 0.096$ because $\hat{q} = 45 \leq M = 50 \leq \bar{q} = 97$.*

Note that, a constraint may fail even though its possibility achieves 1 and holds even though its necessity is 0. In addition, an often used definition says that the possibility measure Π gives always the best case and shows the most optimist attitude, while the necessity N gives the worst case and shows the most pessimist attitude. Then, as presented above, Π and N are related to each others by a dual relationship. Therefore, a combination of these two measures allows the expression of both optimistic

and pessimistic attitude of the decision maker. From these remarks, we can conclude that it is more efficient at this step to use the linear combination of possibility and necessity measures proposed by Brito et al. [4], rather than treating each measure separately. This linear combination is defined as the following:

Given a constraint A , its weight denoted by $W(A)$ which corresponds to the combination of the weighted possibility and necessity, is expressed by:

$$W(A) = \lambda \Pi(A) + (1 - \lambda) N(A) \geq \alpha. \quad (2.11)$$

where the parameter $\lambda \in [0, 1]$, measures the degree of optimism or confidence of the decision maker such that:

$$\lambda = \begin{cases} 1 & \text{Total optimistic case} \\ 0 & \text{Total pessimistic case} \\ 0.5 & \text{Neither optimistic nor pessimistic.} \end{cases} \quad (2.12)$$

and $\alpha \in [0, 1]$ is a given threshold of satisfaction fixed by the decision maker. This formula indicates that the weight measure $W(A)$ must be higher than a given threshold α . The higher it is, the greater the constraint will be satisfied.

Secondly, knowing that propagating the uncertainty of multi-objective problem's data through the resolution model leads often to uncertain formulation of objective functions and as in our case the uncertain data are represented by triangular fuzzy numbers, the objective functions will be consequently disrupted by this fuzzy form. Let us assume that, a multi-objective triangular-valued function can be mathematically defined as:

$$F : X \rightarrow Y \subseteq (R \times R \times R)^n, \\ F(x) = \vec{y} = \begin{pmatrix} y_1 = [\underline{y}_1, \hat{y}_1, \overline{y}_1] \\ \dots \\ y_n = [\underline{y}_n, \hat{y}_n, \overline{y}_n] \end{pmatrix} \quad (2.13)$$

Clearly, in this case, the classical multi-objective techniques cannot be applied since they are only meant for deterministic case. Therefore, a need for special optimization methods techniques to handle the generated triangular-valued functions is evident. To this end, we first introduce a new Pareto dominance over triangular fuzzy numbers, in both mono-objective and multi-objective cases.

2.4.3 New Pareto Optimality over Triangular Fuzzy Numbers

In this section, we first present new mono-objective dominance relations between two TFNs. Then, based on these mono-objective dominance, we define a new Pareto dominance between vectors of TFNs, for multi-objective case. Note that, the minimization sense is considered in all our definitions.

2.4.3.1 Mono-Objective Dominance Relations

In the mono-objective case, three dominance relations over triangular fuzzy numbers are defined: Total dominance (\prec_t), Partial strong-dominance (\prec_s) and Partial weak-dominance (\prec_w).

Definition 1 Total Dominance

Let $y = [\underline{y}, \hat{y}, \bar{y}] \subseteq \mathbb{R}$ and $y' = [\underline{y}', \hat{y}', \bar{y}'] \subseteq \mathbb{R}$ be two triangular fuzzy numbers. y dominates y' totally or certainly (denoted by $y \prec_t y'$) if: $\bar{y} < \underline{y}'$.

This dominance relation represents the fuzzy disjoint situation between two triangular fuzzy numbers and it imposes that the upper bound of y is strictly inferior than the lower bound of y' as shown by case (1) in Fig. 2.4.

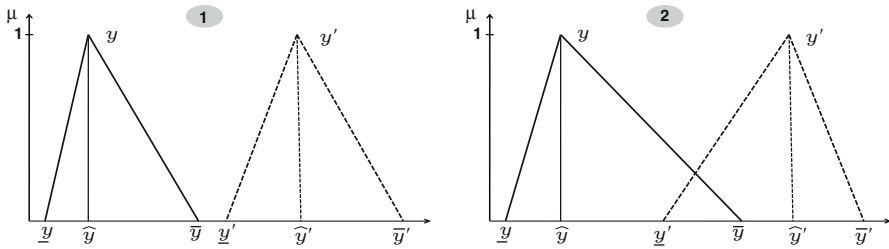


Fig. 2.4 Total dominance and partial strong-dominance

Definition 2 Partial Strong-Dominance

Let $y = [\underline{y}, \hat{y}, \bar{y}] \subseteq \mathbb{R}$ and $y' = [\underline{y}', \hat{y}', \bar{y}'] \subseteq \mathbb{R}$ be two triangular fuzzy numbers. y strong dominates y' partially or uncertainly (denoted by $y \prec_s y'$) if:

$$(\bar{y} \geq \underline{y}') \wedge (\hat{y} \leq \hat{y}') \wedge (\bar{y} \leq \bar{y}').$$

This dominance relation appears when there is a fuzzy weak-overlapping between both triangles and it imposes that firstly there is at most one intersection between them and secondly this intersection should not exceed the interval of their kernel values $[\hat{y}, \hat{y}']$, as shown by case (2) in Fig. 2.4.

Definition 3 Partial Weak-Dominance

Let $y = [\underline{y}, \hat{y}, \bar{y}] \subseteq \mathbb{R}$ and $y' = [\underline{y}', \hat{y}', \bar{y}'] \subseteq \mathbb{R}$ be two triangular fuzzy numbers. y weak dominates y' partially or uncertainly (denoted by $y \prec_w y'$) if:

1. Fuzzy overlapping

$$[(\underline{y} < \underline{y}') \wedge (\bar{y} < \bar{y}')] \wedge [((\hat{y} \leq \underline{y}') \wedge (\bar{y} > \hat{y}')) \vee ((\hat{y} > \underline{y}') \wedge (\bar{y} \leq \hat{y}')) \vee ((\hat{y} > \underline{y}') \wedge (\bar{y} > \hat{y}'))].$$

2. Fuzzy Inclusion

$$(\underline{y} < \underline{y}') \wedge (\bar{y} \geq \bar{y}').$$

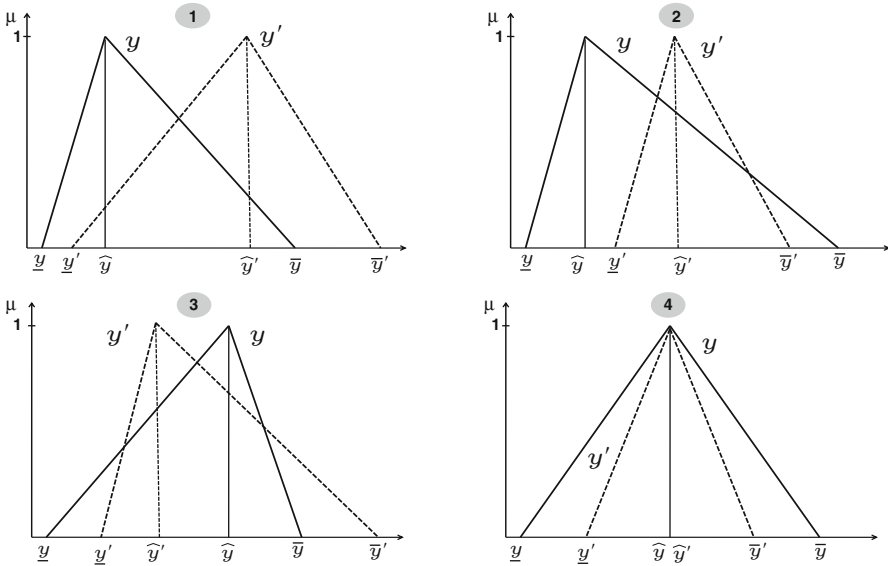


Fig. 2.5 Partial weak-dominance

In this dominance relation, the two situations of fuzzy overlapping and inclusion may occur. Figure 2.5 presents four examples of possible cases, where in (1) and (3) y and y' are overlapped, while, in (2) and (4) y' is included in y .

Yet, the partial weak-dominance relation cannot discriminate all possible cases and leads often to some incomparable situations as for cases (3) and (4) in Fig. 2.5. These incomparable situations can be distinguished according to the kernel value positions in fuzzy triangles. Thus, we propose to consider the kernel values configuration as condition to identify the cases of incomparability, as follows:

$$\hat{y} - \hat{y}' = \begin{cases} < 0, & y \prec_w y' \\ \geq 0, & y \text{ and } y' \text{ can be incomparable.} \end{cases}$$

Subsequently, to handle the identified incomparable situations (with kernel condition $\hat{y} - \hat{y}' \geq 0$), we introduce another comparison criterion, which consists in comparing the discard between both fuzzy triangles as follows:

$$y \prec_w y' \Leftrightarrow (\underline{y}' - \underline{y}) \leq (\overline{y}' - \overline{y})$$

Similarly, it is obvious that: $y' \prec_w y \Leftrightarrow (\underline{y}' - \underline{y}) > (\overline{y}' - \overline{y})$.

It is easy to check that in the mono-objective case, we obtain a total pre-order between two triangular fuzzy numbers, contrarily to the multi-objective case, where the situation is more complex and it is common to have some cases of indifference.

2.4.3.2 Pareto Dominance Relations

In the multi-objective case, we propose to use the mono-objective dominance relations, defined previously, in order to rank separately the triangular fuzzy solutions of each objective function. Then, depending on the types of mono-objective dominance

founded for all the objectives, we define the Pareto dominance between the vectors of triangular fuzzy solutions. In this context, two Pareto dominance relations: *Strong Pareto dominance* (\prec_{SP}) and *Weak Pareto dominance* (\prec_{WP}) are introduced.

Definition 4 *Strong Pareto Dominance.*

Let \vec{y} and \vec{y}' be two vectors of triangular fuzzy numbers. \vec{y} strong Pareto dominates \vec{y}' (denoted by $\vec{y} \prec_{SP} \vec{y}'$) if:

- (a) $\forall i \in 1, \dots, n : y_i \prec_t y'_i \vee y_i \prec_s y'_i$
- (b) $\exists i \in 1, \dots, n : y_i \prec_t y'_i \wedge \forall j \neq i : y_j \prec_s y'_j$
- (c) $\exists i \in 1, \dots, n : (y_i \prec_t y'_i \vee y_j \prec_s y'_j) \wedge \forall j \neq i : y_j \prec_w y'_j$.

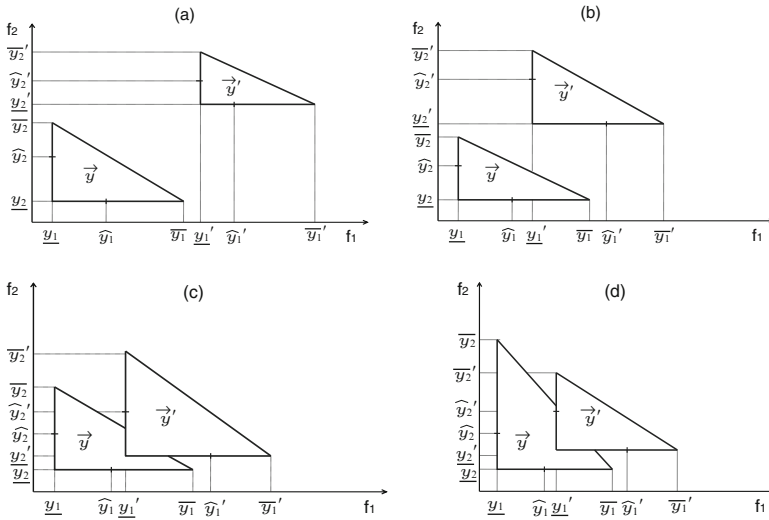


Fig. 2.6 Strong Pareto dominance

The strong Pareto dominance holds if either y_i total dominates or partial strong dominates y'_i in all the objectives (Fig. 2.6a: $y_1 \prec_t y'_1$ and $y_2 \prec_t y'_2$), either y_i total dominates y'_i in one objective and partial strong dominates it in another (Fig. 2.6b: $y_1 \prec_s y'_1$ and $y_2 \prec_t y'_2$), or at least y_i total or partial strong dominates y'_i in one objective and weak dominates it in another (Fig. 2.6c, d: $y_1 \prec_s y'_1$ and $y_2 \prec_w y'_2$).

Definition 5 *Weak Pareto dominance*

Let \vec{y} and \vec{y}' be two vectors of triangular fuzzy numbers. \vec{y} weak Pareto dominates \vec{y}' (denoted by $\vec{y} \prec_{WP} \vec{y}'$) if: $\forall i \in 1, \dots, n : y_i \prec_w y'_i$.

The weak Pareto dominance holds if y_i weak dominates y'_i in all the objectives (Fig. 2.7a). Yet, a case of indifference (defined below) can occur if there is a weak dominance with inclusion type in all the objectives (Fig. 2.7b).

Definition 6 *Case of Indifference*

Two vectors of triangular fuzzy numbers are indifferent or incomparable (denoted by $\vec{y} \parallel \vec{y}'$) if: $\forall i \in 1, \dots, n : y_i \subseteq y'_i$.

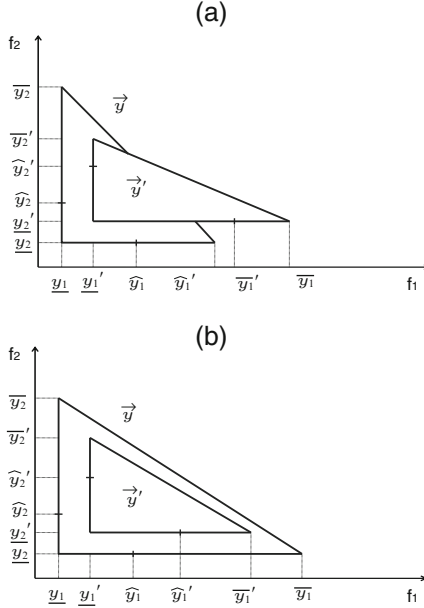


Fig. 2.7 (a) Weak Pareto dominance and (b) Case of indifference

The proposed Pareto dominance in bi-dimensional objective space can easily be generalized for ranking more than two objectives. Note that, if the considered triangular objectives are non-independent, the estimation in a bi-dimensional space can have different distributions (non-triangular) like linear shapes. Finally, the issue now is how integrate this dominance in the research process of multi-objective optimization algorithms.

2.4.4 Extended Optimization Algorithm

In the following, we present an extension of two well-known Pareto-based multi-objective evolutionary algorithms: SPEA2 [36] and NSGAI [9], in order to enable them handling a multi-objective problem with triangular-valued objectives. Both algorithms have proved to be very powerful tools for multi-objective optimization. Due to their population-based nature, they are able to generate multiple optimal solutions in a single run with respect to the good convergence and diversification of obtained solutions. We call our two extended algorithms respectively, ESPEA2—Strength Pareto Evolutionary Algorithm2 and ENSGAI—Non-dominated Sorting Genetic Algorithm II.

2.4.4.1 ESPEA2

SPEA2 is an improved version of the Strength Pareto Evolutionary Algorithm SPEA initially proposed by Zitzler and Thiele [35]. This evolutionary algorithm uses mainly three techniques: a dominance based approach as fitness assignment strategy, a nearest neighbor technique that allows a good diversity preservation and an archive with fixed size that guarantees the elitist storage of optimal solutions. To extend such techniques to triangular fuzzy context, we propose firstly to replace the classical dominance approach by the new Pareto dominance approach proposed for ranking triangular-valued objectives. Secondly, an adaptation of the nearest neighbor technique is introduced. Indeed, in SPEA2, this technique is based on *Euclidean distance* to estimate the density in its neighborhood and it consists in calculating for each solution (objective vector) the distance to its k-nearest neighbor and then adding the reciprocal value to the fitness vector. Yet, as in our case the solutions are triangular objective vectors and knowing that the *Euclidean distance* should be applied only between two exact vectors, we propose to use the expected value as a defuzzification method [32] in order to approximate the considered triangular vectors, such that for each triangular fuzzy number $y_i = [\underline{y}_i, \hat{y}_i, \overline{y}_i]$, the expected value is defined by:

$$E(y_i) = (\underline{y}_i + 2 \times \hat{y}_i + \overline{y}_i) / 4 \quad (2.14)$$

Then, the Euclidean distance between two triangular vectors $\vec{y} = (y_1, \dots, y_n)$ and $\vec{y}' = (y'_1, \dots, y'_n)$ can be applied as follows:

$$D(\vec{y}, \vec{y}') = D(E(\vec{y}), E(\vec{y}')) = \sqrt{\sum_{i=1..n} (E(y_i) - E(y'_i))^2} \quad (2.15)$$

Finally, we adapt the SPEA2 archive to triangular space in order to enable it keeping the obtained triangular solutions. These extensions are integrated into the research process of SPEA2 by modifying the following steps:

- Evaluation: Rank individuals using the new Pareto dominance \prec_{TP} .
- Environmental selection:
 1. Copy all non-dominated individuals having fitness values lower than one in the triangular archive A with fixed size N.
 2. if A is too large ($\text{size}(A) > N$) then, reduce A by means of truncation operator based on Nearest neighbor method to keep only the non-dominated individuals with good spread.
 3. else if A is too small ($\text{size}(A) < N$) then, fill A with the best dominated individuals.
 4. otherwise ($\text{size}(A) = N$), the environmental selection is completed.
- Mating selection: Perform binary tournament selection with replacement on the archive A in order to fill the mating pool.

2.4.4.2 ENSGAI

NSGAI is an extension of an elitism PMOEA called Non-dominated Sorting Genetic Algorithm NSGA, originally proposed by Deb and Srinivas [8]. Unlike the SPEA2 algorithm, NSGAI uses a crowded-comparison operator as diversity preservation technique in order to maintain a uniformly spread front by front. In addition, it does not use an explicit archive for the elitism operation, it only consider the population as a repository to store both elitist and non-elitist solutions. To extend NSGAI to triangular context, we propose at the first step to use the new Pareto dominance between triangular-valued objectives in order to ensure the fitness assignment procedure, in which a dominance depth strategy is applied. At the second stage, we provide an adaptation of the crowded-comparison operator. Indeed, this operator uses the *Crowding Distance* that serves to get a density estimation of individuals surrounding a particular individual in the population. More precisely, the total *Crowding Distance* CD of an individual is the sum of its individual objectives' distances, that in turn are the differences between the individual and its closest neighbors. For the i th objective function y_i , this distance is expressed by:

$$CD(i) = \sum_{i=1..n} (f_{y_i}(i+1) - f_{y_i}(i-1)) / (f_{y_i}^{max} - f_{y_i}^{min}) \quad (2.16)$$

Where f_{y_i} is the fitness value of its neighbors $(i-1)$ and $(i+1)$, $f_{y_i}^{max}$ and $f_{y_i}^{min}$ are respectively the maximum and minimum value of y_i .

However, as in our case, the objective functions are represented by triangular fuzzy values, we propose also to approximate these triangular numbers by calculating their expected values (Eq. (2.14)) before applying the Crowding distance. Finally, it is necessary to adapt both Evaluation and Selection steps in NSGAI, like in SPEA2 algorithm. The distinctive features of NSGAI lie in using the crowding comparison procedure as truncation operator to reduce the population in the environmental selection step and also in considering it as a second selection criteria when two solutions have the same rank in the tournament selection step.

2.5 Application on a Multi-Objective Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is an important combinatorial optimization problem, widely used in a large number of real-life applications [31]. The classical VRP consists in finding optimal routes used by a set of identical vehicles, stationed at a central depot, to serve a given set of customers geographically distributed and with known demands. Through the years, many variants and models derived from the basic VRP have been discussed and examined in the literature. In this work, we are interested in a well-known variant of VRP, the so-called Multi-objective VRP with Time Windows and Uncertain Demands (MO-VRPTW-UD). This variant is

based firstly on the principle of classical VRP, where all the data are deterministic, excepting the customer demands which are uncertain, meaning that the actual demand is only known when the vehicle arrives at the customer location. Several researchers have tried to solve this problem and proposed to deal with the uncertainty of demands using different ways such as probability distributions, dempster belief functions and possibility distributions [1, 13, 28]. In our case, the uncertainty of demands is represented via triangular fuzzy numbers (defined previously) and the objectives to be optimized are respectively, the minimization of the total traveled distance and the total tardiness time.

Formally, a MO-VRPTW-UD may be defined as follows:

Let $G(N,A)$ be a weighted directed graph with an arc set A and a node set $N_i = \{N_0, \dots, N_n\}$ where the node N_0 is the central depot and the other nodes $N_i \neq N_0$ represent the customers. For each customer is associated an uncertain demand dm_i . Only one vehicle k with a limited capacity Q , is allowed to visit each customer. A feasible vehicle route R is represented by the set of served customers, starting and ending at the central depot: $R_k = (N_0, N_1, \dots, N_n, N_0)$. X_{ij}^k denotes the decision variable which is equal to 1 if the vehicle k travels directly from node N_i to node N_j and to 0 otherwise. d_{ij} denotes the symmetric traveled distance between two nodes (N_i, N_j) . This distance is proportional to the corresponding travel time t_{ij} . Figure 2.8 illustrates an example of MO-VRPTW-UD, with a central depot, three vehicles ($V1, V2, V3$) having a maximum capacity $Q = 10$ and a set of eight customers represented by nodes. Each customer $i = 1 \dots 8$ has an uncertain demand expressed in our case by a triangular fuzzy number $dm = [\underline{dm}_i, \widehat{dm}, \overline{dm}]$ (Ex: the fuzzy demand of the customer 1 is $dm_1 = [2, 7, 11]$).

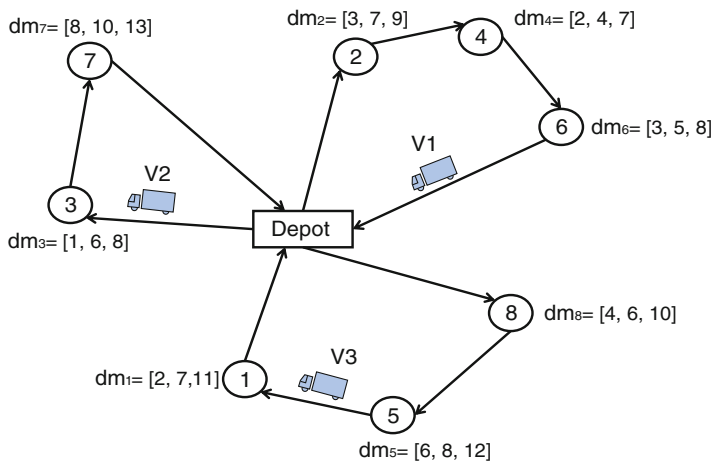


Fig. 2.8 Example of Mo-VRPTW-UD

The main constraints of this problem are: Vehicle capacity constraint, Distance constraint and Time windows constraint.

(1) Vehicle capacity constraint

This constraint imposes that the sum of customer demands in each route must not exceed the limited capacity of associated vehicle. It may be defined as: $\sum_{i=1}^n dm_{N_i} \leq Q$. Yet, as in our case the customers demands are fuzzy values $\widetilde{dm} = [\underline{dm}, \widehat{dm}, \overline{dm}]$, we cannot directly verify if the capacity constraint is satisfied or not and so clearly the constraint satisfaction changes to fuzzy. For example, consider the customer 7 with fuzzy demand $dm_7 = [8, 10, 13]$ shown in Fig. 2.8, we cannot check if dm_7 is lower, equal or higher than $Q = 10$ in order to estimate the transportation costs in terms of time spent and traveled distance. Thus, to handle the satisfaction of this constraint, we propose to use firstly the two measures Π and N of fuzzy constraint satisfaction defined previously. For this example, we obtain $\Pi(dm \leq Q) = 1$ and $N(dm \leq Q) = 0$. Then, by applying the linear combination given by Eq. (2.11) (with for example $\lambda = 0.5$ and $\alpha = 0.2$), we can conclude that the satisfaction of the fuzzy capacity constraint is possible ($W(dm \leq Q) = 0.5 > 0.2$).

(2) Distance constraint

This constraint imposes that each vehicle with a limited capacity Q must deliver goods to the customers according to their uncertain demands dm , with the minimum transportation costs in term of traveled distance. In other words, if the capacity constraint of a vehicle is not satisfied, the delivery fails and causes wasted costs. Therefore, to calculate the traveled distance on a route R defined a priori, three different situations may be found:

- *The demand of a customer is lower than the vehicle capacity ($\sum_{i=1}^f dm_{N_i} < Q$):*
In this case, the vehicle will serve the current customer f and then move to the next one ($f + 1$).
- *The demand of a customer is equal to the vehicle capacity ($\sum_{i=1}^f dm_{N_i} = Q$):*
In this case, the priori optimization strategy is used. In fact, the vehicle leaves the depot to serve the first customer f with its total capacity. As it becomes empty, this vehicle will return to the depot to load and serve the next customer ($f + 1$). Thus, the traveled distance will be: $D(R) = d_{N_0N_1} + \sum_{i=1}^{f-1} d_{N_iN_{i+1}} + d_{N_fN_0} + d_{N_0N_{f+1}} + \sum_{i=f+1}^{n-1} d_{N_iN_{i+1}} + d_{N_nN_0}$.
- *The demand of a customer is higher than the vehicle capacity ($\sum_{i=1}^f dm_{N_i} > Q$):*
In this case, the vehicle will serve the customer f with its total capacity ($Q - \sum_{i=1}^{f-1} d_{N_i}$), go to the depot to load, return back to the same customer f to deliver the remaining quantity and then move to the next customer ($f + 1$). Thus, the traveled distance will be: $D(R) = d_{N_0N_1} + \sum_{i=1}^{n-1} d_{N_iN_{i+1}} + d_{N_fN_0} + d_{N_0N_f} + d_{N_nN_0}$.

Yet as in our case the demands are represented by a triplet of fuzzy values, we propose to calculate separately the distance for each value of the triangular fuzzy demand based on the three situations presented above. Consequently, the traveled distance will be calculated three times and so obtained as triangular number $D = [\underline{D}, \widehat{D}, \overline{D}]$.

(3) Time windows constraint

This constraint imposes that each customer will be served within its time window that represents the interval of time planned for receiving the vehicle service. This means that, if the vehicle arrives too soon, it should wait until the arrival time of its time window to serve the customer, while if it arrives too late (after the fixed departure time), wasted cost in term of tardiness time appears. The time windows constraint uses the following notations:

- The central depot has a time window $[0, l_0]$, meaning that each vehicle that leaves the depot at a time 0 goes back to the depot before the time l_0 .
- Each customer i will be served within his time window $[e_i, l_i]$ by exactly one vehicle, where the lower bound e_i represents the earliest arrival time and the upper bound l_i represents the latest arrival time for the visit of vehicles.
- A waiting time W_i means that the vehicles must arrive before the lower bound of the window e_i .
- A_i, B_i refers respectively to the arrival and the departure times to the customer i .
- Each customer imposes a service time S_i^k that corresponds to the goods loading/unloading time used by the vehicle.
- t_{ij} refers to the travel time from customer i to j .

Firstly, the time needed to serve two consecutive customers i and j is defined as follows:

$$x_{ij}^k(S_i^k + t_{ij} + S_j^k) \leq 0 \quad \text{with} \quad A_i + S_i^k \leq B_i.$$

Besides, a vehicle must arrive at a customer i between the time window $[e_i, l_i]$, but if it arrives before the lower bound e_i , it must wait a while W_i . This waiting time is calculated as follows:

$$W_i = \begin{cases} 0 & \text{if } A_i \geq e_i \\ e_i - A_i & \text{otherwise.} \end{cases}$$

where, the arrival time at customer i is equal to: $A_i = B_{i-1} + t_{i,i-1}$ and the departure time is equal to: $B_i = A_i + W_i + S_i$. While, if the vehicle arrives at a customer i after the upper bound of its time window l_i , a tardiness time must be calculated as follows:

$$T_i = \begin{cases} 0 & \text{if } A_i \leq l_i \\ A_i - l_i & \text{otherwise.} \end{cases}$$

In the case of routes failure, wasted costs in term of tardiness time will appear. Yet, knowing that the travel time depends mainly on the traveled distance and as in our case the obtained distance is a triangular value, the time spent to serve customers will be disrupted by this triangular form and consequently the tardiness time will be

also obtained as triangular fuzzy number $T = [\underline{T}, \hat{T}, \overline{T}]$. Finally, all these constraints combined with the constraints of classical VRP model the MO-VRPTW-UD problem (Fig. 2.8).

To solve the MO-VRPTW-UD problem, the two extended SPEA2 and NSGAI algorithms based on our new Pareto optimality are applied. These algorithms are implemented with the version 1.3-beta of ParadisEO under Linux, especially with the ParadisEO-MOEO module dedicated to multi-objective optimization [3]. Subsequently, to validate the proposed algorithms, we choose to test our VRP application using the Solomon's benchmark, which is considered as a basic reference for the evaluation of several VRP resolution methods [27]. More precisely, six different Solomon's instances are used in our experimentation, namely, C101, C201, R101, R201, RC101 and RC201. Yet, in these instances, all the input values are exact and so the uncertainty of customer demands is not taken into account. At this level, we propose to generate for each instance the triangular fuzzy version of crisp demands in the following manner. Firstly, the kernel value (\widehat{dm}) for each triangular fuzzy demand dm is kept the same as the current crisp demand dm_i of the instance. Then, the lower (\underline{dm}) and upper (\overline{dm}) bounds of this triangular fuzzy demand are uniformly sampled at random in the intervals $[50\%dm, 95\%dm]$ and $[105\%dm, 150\%dm]$, respectively. This fuzzy generation manner ensures the quality and reliability of generated fuzzy numbers. Finally, each of the six sampled fuzzy instances is tested on the both algorithms executed 30 times. Since 30 runs have been performed on each algorithm SPEA2 and NSGAI, we obtained for each instance, 30 sets of optimal solutions that represent the Pareto fronts of our problem. Each solution shows the lowest traveled distance and tardiness time, which are represented by triangular numbers. Examples of two Pareto fronts obtained for one execution of the instance C101 using each algorithm are shown in Figs. 2.10 and 2.11, where the illustrated fronts are composed by a set of triangles, such that each triangle represents one Pareto optimal solution. For instance, the bold triangular (in Fig. 2.10) represents an optimal solution with minimal distance (the green side) equal to $[2413, 2515, 2623]$ and tardiness time (the red side) equal to $[284, 312, 295, 280, 315, 322]$. Note that, both algorithms converge to optimal fronts approximation in a very short run-time (Approx. 0.91 min for SPEA2 and 2.30 min for NSGAI). However, we cannot compare results with the obtained results of other proposed approaches for solving MO-VRPTW-UD because of incompatibilities between the objectives to be optimized.

To assess the performance of our both algorithms, we propose to use two well-known unary quality indicators:

(i) *Hypervolume Indicator* (I_H) [38], considered one of the few indicators that measures the approximation quality in terms of convergence and diversity simultaneously. This intuitive quality indicator needs the specification of a reference point Z^{Max} that denotes an upper bound over all the objectives and a reference set Z_N^* of non-dominated solutions. In our case, the quality of a given output set A in comparison to Z_N^* is measured using the Hypervolume difference metric I_H^- . As shown in Fig. 2.9, this indicator computes the difference between these two sets by measuring the portion of the objective space weakly dominated by Z_N^* and not by A .

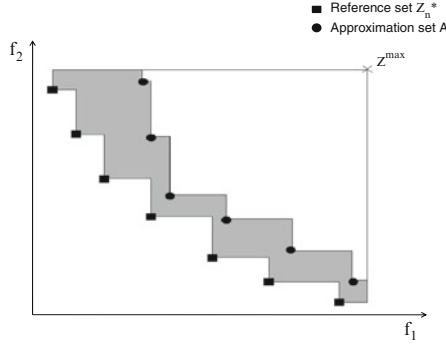


Fig. 2.9 Hypervolume difference indicator

(ii) *Epsilon Indicator* (I_ε) [37], dedicated to the measure of approximations quality in term of convergence. More explicitly, this indicator is used to compare non-dominated approximations and not the solutions. In our case, we use the additive ε -indicator ($I_{\varepsilon+}$) which is a distance based indicator that gives the minimum factor by which an approximation A has to be translated in the criterion space to weakly dominate the reference set Z_N^* . This indicator can be defined as follows:

$$I_{\varepsilon+}^1(A) = I_{\varepsilon+}(A, Z_N^*) \quad (2.17)$$

where

$$I_{\varepsilon+}(A, B) = \min\{\forall z \in B, \exists z' \in A : z'_i - \varepsilon \leq z_i, \forall 1 \leq i \leq n\} \quad (2.18)$$

However, these two indicators are only meant to evaluate the quality of deterministic Pareto front approximations. Thus, to enable them evaluating our uncertain approximations (i.e., Triangular fuzzy solutions), we propose to consider the expected values of the triangular solutions (Function) as the sample of values to be used for the qualification of our both algorithms. In other words, the both indicators are simply applied on the samples of expected values computed for each instance. Therefore, as in our case 30 runs per algorithm have been performed, we obtain 30 Hypervolume differences and 30 epsilon measures for each tested sample. Once all these values are computed, we need to use statistical analysis to be able to compare our two algorithms. To this end, we choose to use Wilcoxon statistical test described in [37].

Table 2.2 gives a comparison of SPEA2 and NSGAII algorithms for the six tested instances. This comparison based on the results of I_H^- and $I_{\varepsilon+}$ indicators, shows that the SPEA2 algorithm is significantly better than the NSGAII algorithm on all the instances, excepting the instances R201 and RC201, where for I_H^- there is no significant difference between the approximations of both algorithms.

Table 2.2 Algorithms comparison using Wilcoxon test with a P -value=0.5%

Instances	Algorithms	I_H^-		$I_{\epsilon+}$	
		SPEA2	NSGAI	SPEA2	NSGAI
C101	SPEA2	-	λ	-	λ
	NSGAI	γ	-	γ	-
C201	SPEA2	-	λ	-	λ
	NSGAI	γ	-	γ	-
R101	SPEA2	-	λ	-	λ
	NSGAI	γ	-	γ	-
R201	SPEA2	-	≡	-	λ
	NSGAI	≡	-	γ	-
RC101	SPEA2	-	λ	-	λ
	NSGAI	γ	-	γ	-
RC201	SPEA2	-	≡	-	λ
	NSGAI	≡	-	γ	-

According to the metric under consideration (I_H^- or $I_{\epsilon+}$), either the algorithm located at a specific row is significantly better (\leftarrow) than the algorithm located at a specific column, either it is worse (\rightarrow) or there is no significant difference between both (\equiv)

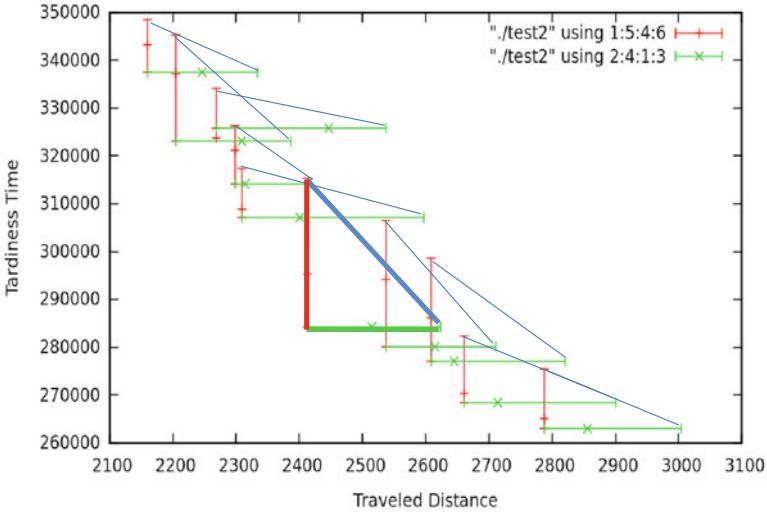


Fig. 2.10 Pareto front (C101-SPEA2)

2.6 Conclusion

This chapter addresses the multi-objective problems with fuzzy data, in particular, with triangular-valued objective functions. To solve such problems, we have proposed an extension of two multi-objective evolutionary algorithms: SPEA2 and NSGAI by integrating a new triangular Pareto dominance. The implemented algorithms have been applied on a multi-objective vehicle routing problem with uncertain demands and then experimentally validated on the Solomon’s benchmark.

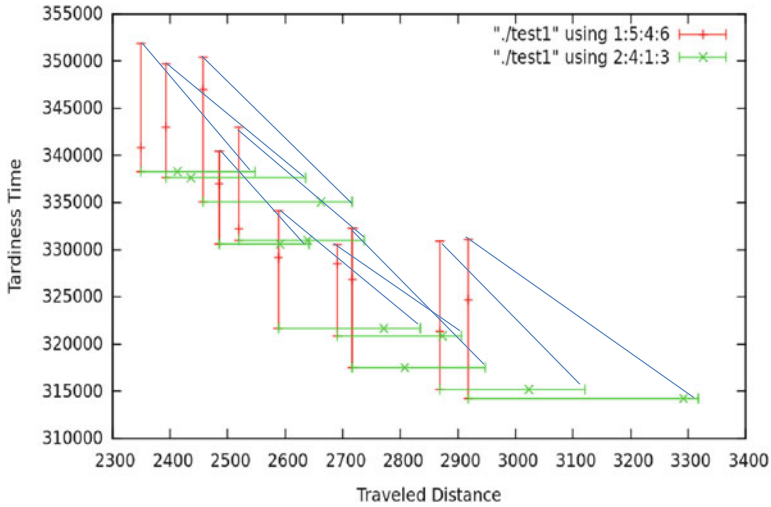


Fig. 2.11 Pareto front (C101-NSGAI)

Subsequently, we have obtained an encouraging results. As a future work, we intend to refine the algorithmic features by introducing for example a new fuzzy distance for the density estimation techniques and to extend the proposed Pareto dominance for ranking other fuzzy shapes like trapezoidal fuzzy numbers. Another perspective will be the extension of multi-objective performance metrics to uncertain context (i.e, fuzzy context).

References

1. S. Asma, E.-G. Talbi, M. Aider, A. Liefoghe, Multi-objective local search with epistemic uncertainty: application to multi-objective vehicle routing problem with uncertain demands, in *ISOR'11* (2011), pp. 1–22
2. M. Babbar, A. LakshmiKantha, D.E. Goldberg, A modified NSGA-II to solve noisy multiobjective problems, in *Genetic and Evolutionary Computation Conference (GECCO'03)*. Lecture Notes in Computer Science, Chicago, IL (Springer, Berlin, 2003), pp. 2723–2727
3. M. Basseur, A. Liefoghe, L. Jourdan, E.-G. Talbi, ParadisEO-MOEO: a framework for evolutionary multi-objective optimization, in *Evolutionary Multi-Criterion Optimization* (Springer, Berlin, 2007), pp. 386–400
4. J. Brito, J.A. Morino, J.L. Verdegay, Fuzzy optimization in vehicle routing problems, in *IFSA-EUSFLAT* (2009)
5. C.A.C. Coello, G.B. Lamont, *Applications of Multi-objective Evolutionary Algorithms* (World Scientific, Singapore, 2004)
6. C.A.C. Coello, G.B. Lamont, D.A. Van Veldhuisen, *Evolutionary Algorithms for Solving Multi-objective Problems* (Springer, New York, 2007)
7. K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms* (Wiley, New York, 2001)

8. K. Deb, N. Srinivas, Multiobjective optimization using nondominated sorting in genetic algorithms. *IEEE Trans. Evol. Comput.* **2**(3), 221–248 (1994)
9. K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2000)
10. D. Dubois, H. Prade, An introductory survey of possibility theory and its recent developments. *J. Jpn. Soc. Fuzzy Theory Syst.* **10**, 21–42 (1998)
11. C.M. Fonseca, P.J. Fleming, Genetic algorithms for multiobjective optimization: formulation, discussion and generalization, in *Proceedings of the Fifth International Conference on Genetic Algorithms* (1993), pp. 416–423
12. C.K. Goh, K.C. Tan, *Evolutionary Multi-objective Optimization in Uncertain Environments: Issues and Algorithms* (Springer, Heidelberg, 2009)
13. G. Goncalves, T. Hsu, J. Xu, Vehicle routing problem with time windows and fuzzy demands: an approach based on the possibility theory. *Int. J. Adv. Oper. Manage. Inderscience* **4**, 312–330 (2009)
14. J. Horn, N. Nafpliotis, D.E. Goldberg, A niched Pareto genetic algorithm for multiobjective optimization, in *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, vol. 1 (1994), pp. 82–87
15. E. Hughes, Evolutionary multi-objective ranking with uncertainty and noise, in *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, London* (Springer, Berlin, 2001), pp. 329–343
16. E.J. Hughes, Constraint handling with uncertain and noisy multi-objective evolution, in *Proceedings of the 2001 Congress on Evolutionary Computation*, Seoul (2001)
17. Y. Jin, J. Branke, Evolutionary optimization in uncertain environments: a survey. *IEEE Trans. Evol. Comput.* **9**(3), 303–317 (2005)
18. J.D. Knowles, D.W. Corne, Approximating the nondominated front using the Pareto archived evolution strategy. *IEEE Trans. Evol. Comput.* **8**(2), 149–172 (2000)
19. A.N. Kolmogorov, *Foundations of the Theory of Probability*, 2nd edn. (Chelsea Pub Co., New York, 1960)
20. M.H. Laarabi, R. Sacile, A. Boulmakoul, E. Garbolino, Ranking triangular fuzzy numbers using fuzzy set inclusion index, in *Fuzzy Logic and Applications* (Springer, Cham, 2013), pp. 100–108
21. P. Limbourg, Multi-objective optimization of problems with epistemic uncertainty, in *Evolutionary Multi-Criterion Optimization* (Springer, Berlin, 2005), pp. 413–427
22. P. Limbourg, E.S. Daniel, An optimization algorithm for imprecise multi-objective problem functions. *Evol. Comput.* **1**, 459–466 (2005)
23. B. Oumayma, B.A. Nahla, E.-G. Talbi, A possibilistic framework for solving multi-objective problems under uncertainty: definition of new Pareto optimality, in *IPDPSW 2013* (2013), pp. 405–414
24. L.F. Paquete, T. Stutzle, Stochastic local search algorithms for multiobjective combinatorial optimization: methods and analysis, in *Handbook of Approximation Algorithms and Metaheuristics*, vol. 13 (Chapman & Hall/CRC Boca Raton, 2007)
25. G. Shafer, *A Mathematical Theory of Evidence* (Princeton University Press, Princeton, 1976)
26. P. Smets, Constructing the pignistic probability function in a context of uncertainty, in *Proceeding 5th Conference on Uncertainty in Artificial intelligence, Windsor* (1989), pp. 29–40
27. M.M. Solomon, Algorithms for the vehicle routing and scheduling problem with time window constraints. *Oper. Res.* **35**(2), 254–265 (1987)
28. D. Sulieman, L. Jourdan, E.-G. Talbi, Using multiobjective metaheuristics to solve VRP with uncertain demands, in *IEEE Congress on Evolutionary Computation* (2010), pp. 1–8
29. E.-G. Talbi, *Metaheuristics: From Design to Implementation* (Wiley, New York, 2009)
30. J. Teich, Pareto-front exploration with uncertain objectives, in *Evolutionary Multi-Criterion Optimization (EMO2001)*. Lecture Notes in Computer Science, vol. 1993 (2001), pp. 314–328
31. P. Toth, D. Vigo, *The Vehicle Routing Problem* (SIAM, Philadelphia, 2002)

32. Z. Wang, F. Tian, A note of the expected value and variance of fuzzy variables. *Int. J. Nonlinear Sci.* **9**(4), 486–492 (2010)
33. L.A. Zadeh, Fuzzy sets. *Inf. Control* **16**, 338–353 (1965)
34. L.A. Zadeh, Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets Syst.* **100**, 9–34 (1999)
35. E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.* **3**(4), 257–271 (1999)
36. E. Zitzler, M. Laumanns, L. Thiele, SPEA2: improving the strength Pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Zurich, May 2001
37. E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, D. Grunert, Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Trans. Evol. Comput.* **7**, 117–132 (2003)
38. E. Zitzler, L. Thiele, J.D Knowles, A tutorial on the performance assessment of stochastic multiobjective optimizers, in *Proceeding of the Third International Conference on Evolutionary Multi-Criterion Optimization* (2005)

Chapter 3

Combining Neighborhoods into Local Search Strategies

Renaud De Landtsheer, Yoann Guyot, Gustavo Ospina,
and Christophe Ponsard

Abstract This paper presents a declarative framework for defining local search procedures. It proceeds by combining neighborhoods by means of so-called *combinators* that specify when neighborhoods should be explored, and introduce other aspects of the search procedures such as stop criteria, solution management, and various metaheuristics. Our approach introduces these higher-level concepts natively in local search frameworks in contrast with the current practice which still often relies on their ad-hoc implementation in imperative language. Our goal is to ease the development, understanding, experimentation, communication and maintenance of search procedures. This will also lead to better search procedures where lots of efficiency gains can be made both for optimality and speed. We provide a comprehensive overview of our framework along with a number of examples illustrating typical usage pattern and the ease of use of our framework. Our combinators are available in the search component of the OsaR.cbls solver.

Keywords Local Search • Metaheuristics • Search Strategies • Neighborhoods • Combinators

3.1 Introduction

Local search is a well-established approach for tackling large combinatorial problems within reasonable amounts of time. A local search solver is built on two main components: a model, and a search strategy. The model represents the problem, and includes the variables and constraints that constitute the problem. A *search procedure* specifies how the search will find a proper solution to the problem. It is made of several components such as [8, 23]:

- *Neighborhoods*, which represent sets of “close” solutions that can be reached from the current solution in one *move*. Neighborhoods can be compared on their

R. De Landtsheer (✉) • Y. Guyot • G. Ospina • C. Ponsard
CETIC Research Centre, Charleroi, Belgium
e-mail: rdl@cetic.be; yg@cetic.be; go@cetic.be; cp@cetic.be

varying efficiency, optimality, and connectivity. They can also be composed together to reach new trade-offs around these aspects.

- *Strategies* to escape from local minima, also called metaheuristics, such as tabu search, simulated annealing, random restart, etc. [9, 8].
- *Solution managers*, which allow us to store the best solution found during the search, and restore it when needed.
- *Stop criteria* to identify when the search will not find any more relevant solutions.

A good design of search procedures is critical because it influences both the efficiency of the search, and the quality of the solution. Designing a local search procedure is a tedious work, compared to the corresponding work required e.g. for Constraint Programming (CP) solvers. Furthermore, local search strategies are still often expressed in procedural programming style because of the host programming language (Java, C++) or the habit to use such a paradigm. This negatively impacts the ease of development, tuning and maintenance of search procedures.

This paper presents a framework to facilitate the development of search procedures. The approach relies on a library of so-called *combinators* that proposes the features commonly found in search procedures as standard bricks. The goal of our design is to define a Domain-Specific Language (DSL) capturing the right abstractions for easily specifying search strategies while keeping the relevant tuning knobs available to the developer. Our approach can speed up the development and experimentation of search procedures when developing a specific solver based on local search, and hopefully, enable the developers to be more focused on the considered problem. Besides, using such a high-level language for defining search procedures also makes them easier to be understood, shared and maintained.

An implementation of these combinators is available in the search component of the `OscAR.cbls` solver. `OscAR.cbls` is an open source solver for constraint-based local search [23]. It features a powerful modeling framework including variables of both integer and set of integer types and including roughly ninety invariants and constraints. Using these invariants and constraints, users can formulate their optimization problem declaratively and benefit from generic and efficient model evaluation methods such as partial propagation to efficiently explore neighborhoods [5, 17]. The topic of this paper being search strategies, it focuses on the search component exclusively. Our implementation is written in the Scala language [20] as the rest of `OscAR`. The Scala supports for DSL was used and enables its transposition to other frameworks.

This paper is structured as follows: Sect. 3.2 explains how our contribution compares to existing approaches. Section 3.3 presents the main principles of neighborhood combinators and describes the neighborhoods available in our framework. Section 3.4 shows the combinators in action on a detailed example of a search procedure applied to the uncapacitated warehouse location problem. Section 3.5 presents an

overview of the combinators currently available in our framework. Section 3.6 further illustrates the use of our combinator framework on a routing problem. Finally Sect. 3.7 summarizes the main benefits and current limitations of our framework.

3.2 Related Work

Developing high-level constructs to simplify the development of search strategies is not a new idea. It has been proposed for CP engines as well as local search approaches, with various trade-offs between expressiveness, ease of use, and flexibility.

Generalised Local Search (GLS) Machines introduce the idea of organizing neighborhoods in a state-machine fashion, each state corresponding to a neighborhood, and transitions describing when the strategy should switch between neighborhoods [12]. Our combinators compare to GLS machines in the same way as structured code (if, while, etc.) compare to state machines, that is: they introduce some hierarchical structure and do not allow representing any transitions easily. Our combinators support all components of search procedures described in Sect. 3.1.

EasyLocal++ organizes search strategies around two concepts, namely runners, and solvers [6]. Examples of runners are tabu search and simulated annealing. Solvers control the search by generating the initial solutions, and deciding how, and in which sequence, runners have to be activated. Example of solvers are round-robin (a.k.a. token-ring strategy), and multi-start. Our framework unifies runners and solvers into neighborhoods, which might exhibit very complex behavior including executing a round-robin among several neighborhoods.

Localizer is a modeling language for local search that offers powerful constructs for defining search procedures. These constructs are related to the accepting criterion as well as simple neighborhoods [13]. Our framework supports some of the neighborhoods proposed by localizer, and is also extendable to any kind of neighborhood, provided it is represented as a class with the proper inheritance from our framework. Also, Localizer proposes a standard language for defining accepting criteria, which is also incorporated into our framework.

The Comet system includes two mechanisms to combine neighborhoods by representing moves as closures, and attach some custom code variables by means of events, triggered on value changes [23]. The gain of such high-level approach for specifying search procedures has been demonstrated on concrete examples [22]. Our approach generalizes this philosophy by proposing several standard ways of combining neighborhoods, and extending to other aspects of search procedures such as stop criterion and metaheuristics. The downside is that our approach might seem less flexible, although custom code can be embedded into search procedures through the dedicated combinator.

Some patterns of neighborhood combination tend to appear in scientific publications on local search, notably in the context of scheduling [21].

Paradiseo supports mechanisms to easily specify tabu search, simulated annealing, and hill-climbing [3]. Our framework unifies searches and neighborhoods into a single API, and lets us tune search procedures more precisely.

SPIDER [10] is a graphical language for representing search strategies, by means of *control networks*. This approach focuses mainly on the definition of the neighborhoods to be applied while we focus more on the overall strategy. Our approach favors a textual DSL that can be expressed close to the model and maintained using standard code revision tools.

ToOLS, OPL, and Search Combinators are frameworks that support the declaration of search strategies for backtracking-based engines such as constraint programming, with variants such as LDS [4, 7, 19]. Their approach is very similar to ours, except that our framework is dedicated to local search. As such, our framework supports the declaration of metaheuristics influencing the acceptance function, tolerates a degradation of the objective function (e.g. for tabu searches) and allows representing the cross-product of two neighborhoods as a single combined neighborhood.

In [24], a mechanism called *Constraint combinators* is proposed to combine several constraints in the context of local search engines. Our approach is concerned about combining neighborhoods, and not the constraints in the model.

Localsolver is a commercial solver that implements local search [1]. This solver is a black-box because the search procedure is standard among all models and cannot be customized by the user. We pursue a different goal, namely to ease people defining their own search strategy, and possibly develop innovative ones, dedicated to their own optimization problems.

3.3 Principle of Neighborhood Combinators

In our framework, a neighborhood is represented by a class instance that can be queried for a move, given the current solution, an acceptance criterion, and an objective function. Neighborhood queries return either the message *NoMoveFound* or the message *MoveFound* that carries a description of the move, and the value of the objective function once the move will be committed. The returned move is expected to be acceptable with respect to the given acceptance criterion and objective function. Querying a neighborhood for a move does not commit the move, although it requires a computational exploration of the neighborhood. The global search loop repeatedly queries moves and commits them until some stopping criterion is met, or until no move can be found by the neighborhood.

The result of combining neighborhoods are still neighborhoods, offering this same API. The most intuitive combination of neighborhoods is “*Best*”. Let a and b be neighborhoods, the following statement is also a neighborhood (statements and code fragments are written in Scala [20]):

```
new Best(a, b)
```

When the combined neighborhood above is queried for a move, it queries both a and b for a move. It then returns the move having the lowest value for the objective function, according to the values carried by the returned moves. If a neighborhood cannot find a move, the overall result is given by the other neighborhood. If no neighborhood could find a move, the combined neighborhood does not find a move. Combinators are implemented in our framework as a DSL, enabling the use of a lighter infix notation. The above example can be rewritten as follows:

```
a best b
```

Besides combinators, our framework includes a set of neighborhoods that can be used to develop custom search procedures. These include:

- Standard domain-independent neighborhoods on arrays of integer variables such as *assignNeighborhood* that changes the value of a single decision variable in an array, *swapsNeighborhood* that swaps the value of two decision variables in an array, and *RandomizeNeighborhood* that randomizes the value of a fraction of integer variables in an array, etc.
- Scheduling neighborhoods such as relaxing and flattening the critical path [14].
- Routing neighborhoods such as *one-point-move*, *two-opt*, etc. [11].

Domain-independent neighborhoods are most interesting because they are quite flexible to be used in very different domains. They also include several features including symmetry elimination, mechanisms to perform intensification or tabu search, the possibility to specify whether the best or the first move is required, and hot restarting. A *hot restart* is the possibility to start the neighborhood exploration from the last explored point in the previous query instead of starting from the initial position at each query. Other neighborhood offer similar features.

3.4 Six Shades of Warehouse Location

This section illustrates how a dedicated search procedure can be developed using neighborhood combinators, for solving the uncapacitated warehouse location problem. The goal is to convince that this framework captures the right abstractions for easily specifying search strategies while keeping the relevant tuning knobs available to the developer. For a discussion on the best search procedures for this problem, see e.g. [23].

The *uncapacitated warehouse location problem* takes as input a set of potential warehouses W and a set of stores S . Each warehouse has a fixed cost f_w and the transportation cost from warehouse w to store s is given by c_{ws} . The problem is to find a subset of warehouses to open in order to minimize the sum of the fixed and transportation costs. Each store is assigned to its nearest open warehouse.

We consider the two following neighborhoods:

- *Switching a single warehouse*: either closing an open warehouse, or opening a closed warehouse. This neighborhood is of size $O(\#W)$ and is connected, so each point of the solution space can be reached by this neighborhood.
- *Swapping two warehouses*: simultaneously close an open warehouse and open a closed warehouse. This neighborhood is of size $O(\#W^2)$ and is not connected, some points of the solution space cannot be reached by this neighborhood. However, this neighborhood has more neighbors than the first one.

The first neighborhood ensures that the whole solution space is reachable, and the second enables us to reach a better solution than the first neighborhood would reach alone. Let *warehouseOpenArray* be an array of Boolean variables. The neighborhoods can be instantiated as follows:

```
val switchWarehouse = new AssignNeighborhood(warehouseOpenArray)
val swapWarehouses = new SwapsNeighborhood(warehouseOpenArray)
```

A first idea is to perform moves from *switchWarehouse* until it cannot find more moves, then switch to *swapWarehouses* and look for moves there. Once *swapWarehouses* is exhausted, come back to *switchWarehouse* and so on. This behavior enables us to use the same neighborhood as long as it can find a move because neighborhoods are more efficient when started from their previous end, since they have some ‘hot restart’ mechanism. Besides, switching back to the first neighborhood after the second one is exhausted makes the combined neighborhood connected. This behavior is implemented in the *ExhaustBack* combinator. Through the infix notation, the combined neighborhood is built as follows:

```
val switchEBSwap = switchWarehouse
                    exhaustBack swapWarehouses
```

Let be *obj* an objective function, neighborhoods, including combined ones, support a standard outer search loop, which can be run through a call to the method *doAllMoves* as follows:

```
switchEBSwap.doAllMoves(obj)
```

This method also inputs an optional stop criterion and an acceptance function. They both have default values and are therefore not represented here above. The default stop criterion never stops the search, so search is stopped because the neighborhood has no acceptable neighbor and the default acceptance function only accepts moves that strictly reduce the objective function.

The neighborhood *switchEBSwap* might be trapped into some local minima. To escape from them, we can use a neighborhood that performs a random move when *switchEBSwap* cannot not find any improving move. It can be instantiated from our standard library of neighborhoods as follows:

```
val randomize = new RandomizeNeighborhood(
                    warehouseOpenArray, W/5)
```

This neighborhood receives two inputs, namely the array on which it operates and the number of variables it has to randomize in this array of variables. Here, one fifth of the array will be randomized.

We can incorporate it into our search strategy as follows:

```
val switchEBSwapORRandom = switchEBSwap
                             orElse randomize
```

On each query, the *orElse* combinator forwards the query to its left-hand side neighborhood first, and queries the right-hand side neighborhood only if the left-hand side one did not find any acceptable move.

An intuitive execution trace of *switchEBSwapORRandom* is illustrated in Fig. 3.1. We represent a *moveFound* by a solid dot, and a *noMoveFound* by a hollow dot. The two first neighborhoods are queried alternately until they are both exhausted. When this occurs, the randomization is performed once and the search starts again. The *orElse* combinator, when switching to its right-hand side neighborhood, resets the internal state of its left-hand side neighborhood. As a consequence, *switchWarehouse* is always queried first after a randomization.



Fig. 3.1 Illustrating the behavior of *switchEBSwapORRandom*

switchEBSwapORRandom has a termination issue since the search will always be able to jump away and continue because of the random move. We therefore need to bound it somehow. This is done by bounding the number of random moves. We use the *maxMoves* combinator to this end. This leads us to *switchEBSwapORRandom2*:

```
val switchEBSwapORRandom2 = switchEBSwap
                             orElse (randomize maxMoves 2)
```

An execution trace of *switchEBSwapORRandom2* is illustrated in Fig. 3.2, using the same convention as above. The neighborhood on the right is the composite *randomize maxMoves 2*. It returns *NoMoveFound* on the third query, and would do so for all subsequent queries as well.

switchEBSwapORRandom2 has one more issue: there is no guarantee that the final solution will be the best one that has been found during the whole search. Indeed, performing a search from a new point might deliver a solution of lower quality than the one we are actually jumping from. We therefore need to save the best solution

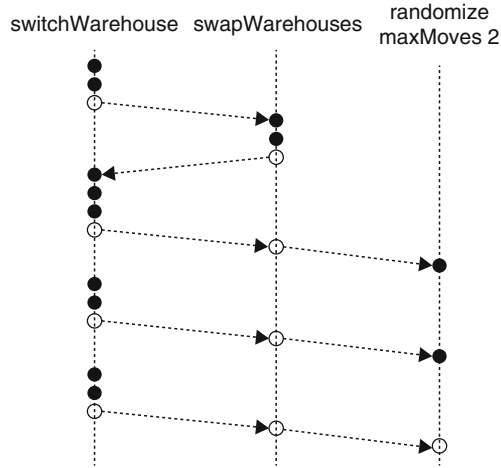


Fig. 3.2 Illustrating the behavior of *switchEBSwapORRandom2*

found, and restore it at some point. This is done by using the *saveBestAndRestoreOnExhaust* combinator. This combinator receives a neighborhood, and an objective function. It saves the model when its combined neighborhood finds a move that might worsen the value of the objective function. This combinator also restores the best found solution when the combined neighborhood is exhausted.

```
val switchEBSwap = switchWarehouse exhaustBack swapWarehouses
val fullSearch = switchEBSwap
                  orElse (randomize maxMoves 2)
                  saveBestAndRestoreOnExhaust obj
```

Based on this first working search strategy, we can proceed with benchmarking to assess the efficiency of the search strategy, and envision alternative searches, potentially faster or more optimal. Thanks to their declarativeness, combinators make it easy to express different search strategies, as illustrated here below.

One might question the usefulness of actually combining two neighborhoods in this context, and revert to a simpler search strategy with only the *switchWarehouse* neighborhood (beside the restart component). It suffices to replace *switchEBSwap* with *switchWarehouse* in the global *fullSearch* search strategy presented above.

One can also consider other ways of combining *swap* and *switch* neighborhoods, by selecting the neighborhood randomly, in a round-robin fashion, or by using a learning combinator, or by selecting the best move from these two neighborhoods. The learning combinator will randomly query its combined neighborhoods with a bias that is adjusted gradually depending on their efficiency, quantified by the gain on the objective function per time unit spent exploring them. These strategies are respectively instantiated with combinators and must be placed within the *fullSearch* presented here above:

```
val switchRandomSwap = switchWarehouse random swapWarehouses
val switchRRSwap = switchWarehouse roundRobin swapWarehouses
```

```
val switchLearningSwap = new Learning (switchWarehouse ,
                                       swapWarehouses)
val switchBestswap = bestSwitchWarehouse best bestSwapWarehouses
```

The last option might not make sense unless both *switchWarehouse* and *swapWarehouses* were instantiated so that they would return their best move instead of their first move. In our framework, this setting is defined at the level of the neighborhood, by specifying this in their respective constructor as shown below:

```
val bestSwitchWarehouse = new AssignNeighborhood (
                                       warehouseOpenArray ,
                                       best=true)
val bestSwapWarehouses = new SwapsNeighborhood (
                                       warehouseOpenArray ,
                                       best=true)
```

Yet another approach is to consider simulated annealing. It can be used on a combination of neighborhoods, such as the search components declared here above. In the search procedure here below, we use it over the *switchWarehouse* neighborhood with a temperature function defined as $\max(0, 100 - \text{“iteration number”})$ and the exponent base set to 2. Notice that the stop criterion is modified for the example; now it automatically stops the search when no improving move has been taken for, say, the last $W/2$ iterations.

```
val globalSimulatedAnnealing =
    switchWarehouse
    metropolis (( it : Int ) => max(0,100 - it) , 2)
    maxMoves W/2 withoutImprovementOver obj
    saveBestAndRestoreOnExhaust obj
```

Different search strategies should of course be benchmarked against typical data sets to select the most adequate one.

3.5 Building a Library of Combinators

This section presents the main combinators of our framework. These combinators are accessible either directly, as Scala classes, or through a DSL in infix notation. The library covers the aspects involved in the search procedures described in Sect. 3.1 plus additional features to compose neighborhood into atomic ones, and embed native code into the strategy. This library can be extended easily, by providing additional implementations of the available *Neighborhood* base class.

3.5.1 Neighborhood and Move Selection Combinators

They select the neighborhood that is actually queried for a move, or the move that is actually returned.

- `OrElse (a : Neighborhood , b : Neighborhood)`
If a finds a move, returns it, otherwise, returns the result of b .
- `RoundRobin (l : List [Neighborhood] , steps : Int)`
Performs a round robin of queries on the neighborhoods in l . Switches to the next neighborhood in the list as soon as the current one is exhausted or has been queried *step* number of times. It rolls back to the first neighborhood after the last one.
- `Exhaust (a : Neighborhood , b : Neighborhood)`
Returns the result of querying a until it cannot find more moves. It then switches to b and never comes back to a except if an explicit reset is done.
- `ExhaustBack (a : Neighborhood , b : Neighborhood)`
Returns the result of querying a until it cannot find more moves. It then switches to b and comes back to a when b cannot find moves, and so on.
- `Random (a : Neighborhood , b : Neighborhood)`
Randomly queries a or b . If the first neighborhood tried cannot find any move, returns the result of the other one.
- `Learning (l : List [Neighborhood] , n : Int)`
Randomly queries a or b with a bias that is adjusted gradually depending on the efficiency of the neighborhoods, quantified by the gain on the objective function per time spent exploring the neighborhoods. The bias is updated every n iteration as the mean between its previous value and the slope of the neighborhoods since the last update of the bias.
- `Sequence (a : Neighborhood , b : Neighborhood)`
Performs one query on a . All next queries are forwarded to b .
- `Best (a : Neighborhood , b : Neighborhood)`
Selects the best move between the ones returned by querying a and b .

3.5.2 Acceptation Function Combinators

They influence the acceptance function that is actually used for move selection. The acceptance function can be specified in the outer search loop, but these combinators replace it by a specific one.

- `WithAcceptanceCriterion (a : Neighborhood ,
 criterion : (Int , Int) => Boolean)`
This combinator overrides the acceptance criterion given to the neighborhood a .
- `AcceptAll (a : Neighborhood)`
This combinator forces any query on a to accept all moves.
- `Metropolis (a : Neighborhood ,
 temperature : Int => Float ,
 base : Float)`

This combinator injects a metropolis acceptance function on a . Metropolis accepts all the moves that improve the objective function. Worsening moves are accepted with probability

$$\text{base}^{(\text{oldObj} - \text{newObj}) / \text{temperature}(\text{iterationNumber})}$$

The iteration number starts at zero and is reset to zero when the combinator is reset.

3.5.3 Solution Management Combinators

They ensure that the search does not forget the best encountered solution.

- `SaveBest(a: Neighborhood, obj: Objective)`
Saves the found solution that reaches the lowest value for obj during the whole search. Saving is performed on a lazy basis, it means that the solution is only saved just before a move returned by a is committed, if this move might decrease the quality of obj . This combinator assumes that the value for the objective function carried by the move refers to obj . To restore this best solution, we can either call the method `restoreSolution` or use the additional combinator `restoreOnExhaust`.
- `RestoreBestOnExhaust(a: SaveBest)`
Automatically restores the best solution saved by a when it cannot find any improving moves. If the current solution is better than the saved one, the latter is not restored.

3.5.4 Stop Criterion Combinators

They specify when search must be stopped. They can be used also to change the search policy.

- `MaxMoves(a: Neighborhood, maxAllowedMove: Int)`
Bounds the number of moves that can be done on neighborhood a . Additional queries on a will always return `NoMoveFound` without exploring a .
- `MaxMovesWithoutImprovement(a: Neighborhood, nbMoves: Int, obj: Objective)`
Bounds the number of moves that can be done on neighborhood a without improvements over the best value of the objective function obj .
- `StopWhen(a: Neighborhood, cond: () => Boolean)`
Prevents any new query on a when `cond` evaluates to `true`.

3.5.5 Code Embedding Combinators

They allow the execution of custom code at different times of the neighborhood exploration.

- `OnMove(a: Neighborhood , proc: Move => Unit)`
Attaches some custom code to the move of *a*, encapsulated in *proc*. This code is executed when the move is committed.
- `OnQuery(a: Neighborhood , proc: () => Unit)`
Executes some custom code, encapsulated in *proc*, when the neighborhood *a* is queried for a move. The code is executed before the query is forwarded to the neighborhood.

3.5.6 Neighborhood Aggregation Combinators

They make possible to propose new neighborhoods e.g. by taking the cross-product of several existing neighborhoods.

- `AndThen(a: Neighborhood , b: Neighborhood ,
maxIntermediaryDegradation: Int)`
This combinator is the cross-product of *a* and *b*, that is: a move of *andThen* is a succession of a move from *a* and a move from *b*. *maxIntermediaryDegradation* is the maximal degradation that is admitted for the intermediary step.
- `Atomic(a: Neighborhood)`
Specifies that the neighborhood *a* cannot be interrupted by any external stop criterion. The exhaustion of this neighborhood is considered as a single move.

3.6 A Vehicle Routing Example with Combinators

This section shows that combinators prove equally expressive in another area, namely vehicle routing problems. In this example, we suppose that the standard routing neighborhoods (*insertPoint*, *onePointMove*, *threeOpt*, *swapInsert*) are available with the proper API, as in the *Oscar.cbls* framework [18].

We consider a problem where a vehicle has a capacity and a depot, and serves orders by delivering the quantity ordered at the delivery point. The vehicle can pass several times by the depot to serve as many orders as possible within a restricted time frame. The objective function considers both the distance and the number of served orders. In the short run, injecting a depot point in a circuit is not a desirable move as it increases the travel distance. A possible way of doing is to force the injection, by accepting moves that worsen the objective. Another strategy, instantiated below, is to inject a depot and one more customer at the same time, so that the couple

is actually a desired move. We consider the cross-product of injecting a depot and serving one more customer as a depot insertion strategy. To represent this strategy, we use the *andThen* combinator. In the search procedure presented here below, we consider two alternating phases: a phase where points and depots are injected into the route, and a phase where the route is optimized by applying standard routing neighborhoods (*onePointMove*, *threeOpt*, *swapInsert*, etc.) combined by the learning combinator that foster the most efficient neighborhood. Since depots are not often needed, we insert customers as much as possible, and try inserting depots when no more customers can be inserted.

```
val routingWithDepotSearch =
  insertPoint
  orElse (insertDepot andThen insertPoint)
  exhaustBack new Learning (onePointMove, threeOpt,
    swapInsert, ...)
```

Again, modifying this search strategy is straightforward. For instance, we might want to improve the quality of the routes before inserting a depot. This might help because a depot is only needed if the capacity is the limiting factor for inserting more customers. We can for instance move the *onePointMove* in an *exhaustBack* fashion next to the *insertPoint*. By performing this one point move search, we have more assurance that the depot insertion will be queried when the limiting factor for inserting more point is the capacity, and not the overall time frame. Similar behavior could also be implemented by using the *stopWhen* combinator on the *andThen*, or by specifying in the *andThen* that the first move cannot violate any strong constraint, through the *maximalIntermediaryDegradation* parameter defined in Sect. 3.5.

```
val routingWithDepotSearch =
  insertPoint exhaustBack onePointMove
  orElse (insertDepot andThen insertPoint)
  exhaustBack new Learning (threeOpt, swapInsert, ...)
```

3.7 Conclusion

This paper presented a declarative framework for easing the development of search procedures for local search solvers. This framework proposes standard building bricks both for defining search procedures elements and for combining them in powerful ways. It was implemented as DSL within the *OscAR* library on top of the *Scala* language and experimented on a set of standard search problems.

Introducing such higher-level constructs will enable the search procedure developers to (1) focus on their specific problem instead of having to consider the specifics of the search loop, (2) experiment more easily with different search procedures, and (3) easily understand, maintain and communicate on search procedures, so our framework might increase the chance of the developers to design better search procedures.

A possible drawback is that genericity often leads to some processing and memory overheads. Our framework is no exception to this: moves are explicitly instantiated before being committed leading to some very small memory overhead, and there are a few additional method calls per move. This run time overhead is negligible, compared to the run time of neighborhood exploration, except for very small neighborhoods that immediately find a proper move. Besides, some of our neighborhoods also suffer from some run time overhead, due to their genericity.

Our framework also imposes some restrictions on the possible search strategies because the whole library of combinators is not a Turing-complete language. Nevertheless, this is not a critical issue because the library can be extended if needed.

Our combinators rely on neighborhoods as basic blocks. We have shown in our example that standard neighborhoods can be used easily, but extending this library will make the overall framework more useable. Potential extensions include neighborhoods on set variables, larger neighborhoods such as ejection chains, constraint-specific neighborhoods, and very large neighborhoods [1, 2, 15].

Our library of combinator itself can be improved. The *AndThen* combinator that builds the cross-product of two neighborhoods could be modified to instantiate the neighborhood on the right dynamically, based on the move explored on the left, e.g. to implement some symmetry elimination in the cross-product.¹ The *Learning* combinator could be given more efficient learning capabilities, etc.

We also hope that combinators might be a good candidate for standardizing how local search procedures can be defined concisely, and be incorporated into standards such as MiniZinc [16]. We also want to propose profiling and benchmarking tools on search strategies to further help developers fine tune their search strategies, just like visualizing the search tree of a CP solver might help developers fine tune the CP search heuristics. These tools will be integrated into the Oscar framework.

To conclude, we want to present a simple, yet actual principle that justifies the use of high-level, declarative, and productive approaches like ours, despite their inherent technical overhead due to their genericity. The principle is that nowadays, to some extent, brain cycle is more expensive and valuable than CPU cycle. We consider that development time is more profitably spent in high-level algorithmic tuning than in the development overhead arising from the use of low-level constructs. The same reasoning let us chose the Scala programming language for the whole Oscar framework instead of, say the C programming language.

Acknowledgements This research was conducted under the SimQRi research project (ERANET CORNET, grant nr 1318172).

¹ Credits to Luca Di Gaspero for the suggestion.

References

1. T. Benoist, B. Estellon, F. Gardi, R. Megel, K. Nouioua, LocalSolver 1.x: a black-box local-search solver for 0-1 programming. *4OR* **9**(3), 299–316 (2011)
2. G. Björndal, J.-N. Monette, P. Flener, J. Pearson, A constraint-based local search backend for MiniZinc, *Constraints*, J. Fast Track CP-AI-OR **20**(3), 325–345 (2015)
3. S. Cahon, N. Melab, E.-G. Talbi, Paradiseo: a framework for the reusable design of parallel and distributed metaheuristics. *J. Heuristics* **10**(3), 357–380 (2004)
4. S. de Givry, L. Jeannin, A unified framework for partial and hybrid search methods in constraint programming. *Comput. Oper. Res.* **33**(10), 2805–2833 (2006)
5. R. De Landtsheer, C. Ponsard, Oscar.cbls: an open source framework for constraint-based local search, in *Proceedings of ORBEL'27*, 2013
6. L. Di Gaspero, A. Schaerf, Easylocal++: an object-oriented framework for the flexible design of local-search algorithms. *Softw. Pract. Exp.* **33**(8), 733–765 (2003)
7. T. Frühwirth, L. Michel, C. Schulte, Constraints in procedural and concurrent languages, in *Constraint Programming Handbook* (Elsevier, Amsterdam, 2006), pp. 451–492
8. M. Gendreau, J.-Y. Potvin, *Handbook of Metaheuristics* (Springer, New York, 2010)
9. F.W. Glover, G.A. Kochenberger, *Handbook of Metaheuristics*. International Series in Operations Research & Management Science (Springer, Berlin, 2003)
10. Emilia GOLEMANOVA, Declarative implementations of search strategies for solving CSPs in control network programming. *WSEAS Trans. Comput.* **12**(4), 174–183 (2013)
11. C. Groer, Parallel and serial algorithms for vehicle routing problems, in *BiblioBazaar*, 2011
12. H.H. Hoos, T. Stützle, *Stochastic Local Search: Foundations and Applications* (Morgan Kaufmann, Burlington, 2005)
13. L. Michel, P. Van Hentenryck, Localizer: a modeling language for local search, in *Principles and Practice of Constraint Programming-CP97*, 1997
14. L. Michel, P. Van Hentenryck, Iterative relaxations for iterative flattening in cumulative scheduling, in *ICAPS*, vol. 4, 2004, pp. 200–208
15. S. Mouthuy, P. Van Hentenryck, Y. Deville, Constraint-based very large-scale neighborhood search. *Constraints* **17**(2), 87–122 (2012)
16. NICTA Optimization Research Group. Minizinc and flatzinc. <http://www.minizinc.org/>
17. Oscar Team, Oscar: Operational research in Scala, 2012. Available under the LGPL licence from <https://bitbucket.org/oscarlib/oscar>
18. C. Ponsard, R. De Landtsheer, Y. Guyot, A high-level, modular and declarative modeling framework for routing problems, in *Proceedings of ORBEL'28*, 2014
19. T. Schrijvers, G. Tack, P. Wuille, H. Samulowitz, P.J. Stuckey, Search combinators, in *Principles and Practice of Constraint Programming* (Springer, Berlin, 2011), pp. 774–788
20. The Scala programming language. <http://www.scala-lang.org>
21. S. Thevenin, N. Zufferey, M. Widmer, Metaheuristics for a scheduling problem with rejection and tardiness penalties. *J. Sched.* **18**(1), 89–105 (2015)
22. P. Van Hentenryck, L. Michel, Control abstractions for local search. *Constraints* **10**(2), 137–157 (2005)
23. P. Van Hentenryck, L. Michel, *Constraint-Based Local Search* (MIT Press, Cambridge, 2009)
24. P. Van Hentenryck, L. Michel, L. Liu, Constraint-based combinators for local search, in *Principles and Practice of Constraint Programming*, 2004

Chapter 4

All-Terrain Tabu Search Approaches for Production Management Problems

Nicolas Zufferey, Jean Respen, and Simon Thevenin

Abstract A metaheuristic is a refined solution method able to find a satisfying solution to a difficult problem in a reasonable amount of time. A local search metaheuristic works on a single solution and tries to improve it iteratively. Tabu search is one of the most famous local search, where at each iteration, a neighbor solution is generated from the current solution by performing a specific modification (called a move) on the latter. The goal of this chapter is to present tabu search approaches with enhanced exploration and exploitation mechanisms. For this purpose, the following ingredients are discussed: different neighborhood structures (i.e., different types of moves), guided restarts based on a distance function, and deconstruction/reconstruction techniques. The resulting *all-terrain* tabu search approaches are illustrated for various production problems: car sequencing, job scheduling, resource allocation, and inventory management.

Keywords Tabu search • Car sequencing • Job scheduling • Resource allocation • Inventory management

4.1 Introduction

As presented in [26], let f be an objective function which has to be minimized. A solution s is optimal for f if there is no better solution than it, that is, there is no solution s' such that $f(s') < f(s)$. As mentioned in [28], an *exact method* guarantees the optimality of the provided solution. However, for a large number of applications and most real-life optimization problems, such methods need a prohibitive amount of time to find an optimal solution, because such problems are NP-hard [7]. For these difficult problems, one should prefer to quickly find a satisfying solution, which is the goal of *heuristic* and *metaheuristic* solution methods. There mainly exist three families of (meta)heuristics: constructive algorithms (a solution is built step by step from scratch, like the greedy algorithm where at each step, the best element is added

N. Zufferey (✉) • J. Respen • S. Thevenin
Geneva School of Economics and Management (GSEM), University of Geneva,
Geneva, Switzerland
e-mail: n.zufferey@unige.ch

to the solution under construction), local search methods (a solution is iteratively modified: this will be discussed below), and evolutionary metaheuristics (a population of solutions is managed, like genetic algorithms and ant algorithms). The reader is referred to [8, 24] for more information on metaheuristics and general guidelines to adapt them.

Only the context of local search methods is considered in this work. A *local search* algorithm starts with an initial solution and tries to improve it iteratively. At each iteration, a modification, called a *move*, of the current solution s is performed in order to generate a neighbor solution s' . Let $N(s)$ denote the set of all neighbor solutions of s . The definition of a move, that is the definition of the *neighborhood* structure N , depends on the considered problem. Popular local search methods are the descent local search, simulated annealing, tabu search and variable neighborhood search. In a descent local search, the best move is performed at each iteration and the process stops when a local optimum is found. Tabu search was first proposed by Fred Glover in the 80's and is nowadays still considered as one of the most efficient local search method. To prevent tabu search from being stuck in a local optimum, when a move is performed, the reverse move is forbidden (i.e., set as tabu) for *tab* (parameter) iterations. A basic pseudo-code for tabu search is presented in Algorithm 1.

Algorithm 1 Tabu search

Generate an initial solution s and set $s^* = s$.

While no stopping criterion is met, do

1. from the current solution s , generate the best non-tabu neighbor solution s' ;
2. forbid the reverse move for *tab* (parameter) iterations;
3. set $s = s'$;
4. if $f(s) < f(s^*)$, set $s^* = s$;

Return the best solution s^* .

In most tabu search algorithms, only one neighborhood structure N is used, no restarts are performed, and no major restructuring of the solution is observed. The goal of this chapter is to present *all-terrain* tabu search approaches able to find: (1) a good balance between exploitation (i.e., the ability to guide the search in the solution space and to take advantage of the problem structure) and exploration (i.e., the ability to visit various zones of the solution space); (2) a good tradeoff between intensification and diversification. For this purpose, the three following ideas are discussed: (a) the use of several neighborhood structures, as a local optimum for a neighborhood structure is not necessarily a local optimum for another; (b) the management of guided restarts relying on a distance function, as diversification actions should be triggered if the potential of the current zone of the solution space becomes poor; (c) the restructuring of a solution relying on deconstruction (for diversification) and reconstruction (for intensification) phases, based on the sequential use of a pool of solutions.

Strongly relying on [12, 20, 23, 25], the above presented ingredients are illustrated for various production problems, namely car sequencing (Sect. 4.2, using guided restarts), job scheduling (Sect. 4.3, using deconstruction/reconstruction phases), resource allocation (Sect. 4.4, using moves of different amplitudes), and inventory management (Sect. 4.5, using two neighborhood structures).

4.2 Smoothing the Production for Car Sequencing

4.2.1 Presentation of the Problem

Nowadays, new constraints known as *smoothing constraints* are attracting a growing attention in the area of job scheduling [3] and in particular for car sequencing problems, where cars must be scheduled before production in an order respecting various constraints (colors, optional equipment, due dates, etc.), while avoiding overloading some important resources. For the car plant, balancing between optional equipment and colors allows to respect customers deadlines and to prevent overloading some resources (machines or employees).

In 2005, the car manufacturer *Renault* proposes a car sequencing problem through the ROADEF 2005 Challenge [19], with real instances involving hundreds of cars. Car families are defined so that two cars of the same family contain the same optional equipment. Each optional equipment i is associated with a N_i/P_i *ratio constraint*, meaning that at most N_i cars with option i can be scheduled in any subsequence of P_i cars, otherwise a penalty occurs. The objective is to minimize a weighted function involving ratio constraint violations and the number of color changes. In [13], a variation of the *Renault* problem is studied, where non-identical parallel machines (or production lines) and eligibility constraints are considered (i.e., a job—or a car—can only be performed on some specific machines). The objective function involves three components: makespan, smoothing costs and setup costs. Another variant of the car sequencing problem is studied here, where the violations of the $2/3$ ratio constraint are penalized as smoothing costs in the objective function, with eligibility and makespan constraints.

A set of n jobs, m non-identical machines and eligibility constraints are considered here. Each job j belongs to one of the g available families and has a processing time p_{ij} depending on the machine i . A solution s contains a production sequence for each machine. The goal consists in minimizing the smoothing cost function $f(s)$, which is the weighted number of times there are three consecutive jobs of the same family in s . In addition, the overall makespan cannot exceed an upper bound UB (but UB is set large enough to easily prevent the rejection of jobs). A small value of UB usually indicates a high occupancy rate of the machines, and as a consequence, the production system will be available sooner for future commands.

4.2.2 Solution Methods

Three different methods are proposed: *GR* (a greedy heuristic), *TS* (a conventional tabu search), and *TSGR* (a tabu search with *guided* restarts, managed with a distance function). The time limit of each algorithm is $T = 15$ min (which is consistent from a practical standpoint). Note that if an algorithm stops before T , it is restarted and the best generated solution is returned to the user.

GR starts from an empty solution s . At each step, it inserts the job j in s which minimizes the augmentation of f , while respecting the eligibility and makespan constraints. Each possible insertion is tested and ties are broken randomly. *GR* stops when all jobs are scheduled.

TS starts from an initial solution given by *GR*, and tries to improve it iteratively by performing the best possible non-tabu move. A *move* is defined as positioning a job somewhere else in the solution (in the same sequence or in the sequence of another machine). Each time a move is performed, it is forbidden (tabu) to move it again for tab iterations, where tab is uniformly generated in interval $[3, 7]$ after each move. Larger values of tab do not allow intensifying the search around the encountered local optima.

In *TSGR*, guided restarts of *TS* are performed as follows, where a *cycle* is defined as an execution of *TS* for $I = 100$ iterations. Larger values of I do not allow enough restarts, whereas smaller values do not allow the method to intensify the search around the given initial solution. In other words, a tradeoff has to be found between diversification (associated with small values of I) and intensification (associated with large values of I). Let $s_b^{(k)}$ (resp. $s_i^{(k)}$) be the best visited (resp. initial) solution in cycle k . The *distance* between two solutions s_1 and s_2 is defined as $dist(s_1, s_2) = \sum_j y_j(s_1, s_2)$, where $y_j(s_1, s_2) = 1$ if job j has the same position index in solutions s_1 and s_2 (for the sequence it belongs to, independently of the machine), and $y_j(s_1, s_2) = 0$ otherwise. Note that the same sequence of jobs can appear on two different machines for s_1 and s_2 , which is consistently measured as equivalent situations by the distance function. Then, at the end of a cycle k , if $dist[s_b^{(k-1)}, s_b^{(k)}] < n/4$ (which roughly corresponds to a structural difference below 25% between the two involved solutions), $s_i^{(k+1)}$ is generated by performing 10 random swap moves on $s_b^{(k)}$ (in order to slightly diversify the search from $s_b^{(k)}$), otherwise $s_i^{(k+1)}$ is generated with *GR*. Note that *swap* moves are defined as exchanging the position index of two jobs on the same machine. This mechanism allows to intensify the search if the two best solutions of two consecutive cycles have a similar structure. Otherwise a diversification action is triggered with a restart.

4.2.3 Experiments

An exact linear formulation relying on CPLEX 12.4 has been tested with a time limit of 10 h on an Intel Quad-core i7 @ 3.4 GHz with 8 GB DDR3 of RAM memory.

CPLEX is only able to solve instances with up to 30 jobs, for which the proposed tabu search approaches are usually able to quickly find optimal solutions. For these reasons, exact methods will not be discussed further.

The instances are derived from the ones presented in [13]. Methods *GR*, *TS* and *TSGR* are compared in Table 4.1. For each instance are first given n , m , UB and f^* , which is the best solution value found by any of the algorithm. The next column indicates the percentage gap between f^* and the best solution value found by *GR* within $T = 15$ min. The last two columns present the same information for *TS* and *TSGR* (but the results are averages over 10 runs with $T = 15$ min). The last row indicates the average gaps for the three methods. It can be observed that: (1) *TS* is much more efficient than *GR*, which shows the relevance of the used moves; (2) *TSGR* significantly outperforms *TS*, which indicates that the proposed way to guide the restarts is powerful, and should be investigated for other problems.

Table 4.1 Results on instances with 100 and 300 cars

n	m	UB	f^*	<i>GR</i> (%)	<i>TS</i> (%)	<i>TSGR</i> (%)
100	4	3604	3005	4.49	5.39	0.00
100	4	3612	3005	4.49	2.70	0.00
100	4	3610	2980	5.37	0.39	0.00
100	4	3632	2850	10.18	4.02	0.70
300	5	9005	960	42.92	8.72	2.08
300	5	9038	895	47.82	7.31	4.02
300	5	9086	870	52.07	5.32	1.38
300	5	9143	730	81.23	12.77	0.68
Average				31.07	5.83	1.11

4.3 A Deconstruction-Reconstruction Method for Job Scheduling

4.3.1 Presentation of the Problem

In the considered job scheduling problem, the production environment consists in a set of parallel and identical machines. Given a set J of n jobs, a subset $J' \subseteq J$ must be selected and scheduled before a global deadline D . The non-selected jobs are rejected. With each job j is associated an integer processing time p_j and a gain g_j (incurred if j is performed). Preemptions are allowed at integer points in time, and some pairs of jobs are incompatible: it should be avoided to perform them at common time slots. A *conflict* occurs if two incompatible jobs are processed during a common time slot (there can be more than one conflict between two jobs). The problem is to find a solution s where each performed job j is given p_j time slots,

and such that the number of conflicts $C(s)$ does not exceed a given upper bound K . Two objectives f_1 (to be maximized) and f_2 (to be minimized) are considered in a lexicographical order (i.e., f_1 is infinitely more important than f_2): $f_1(s)$ is the sum of the gains of completely performed jobs, and $f_2(s)$ is the number of parallel machines used in s .

This problem has applications in continuous flow production where multiple resources are required simultaneously to perform a job. Indeed, *incompatibilities* occur when scarce resources are involved in the production system [1]: two jobs which necessitate a common scarce resource cannot be performed simultaneously (they are incompatible). However, it is assumed here that some additional resources can be mobilized up to a certain budget, and thus up to K conflicts are allowed. Papers on scheduling with incompatibilities include [4, 6, 21] and are often related to the graph multi-coloring problem. In particular, the considered problem is a generalization of the well-known and NP-hard k -coloring problem (if $K = 0$, $D = k$, and $p_j = 1$ for each j). Scheduling with *rejections* is growing field of research, where the decision of acceptance or rejection of an order is integrated with scheduling (see [18] for a comprehensive review). Finally, *preemptions* are used in practical situations where setup times are negligible (e.g., in automated production).

4.3.2 Solution Methods

Five approaches are compared: *GR* (greedy algorithm), *DLS* (descent local search), *TS* (tabu search), *TS^R* (tabu search with restarts), and *DRM* (deconstruction/reconstruction metaheuristic). The time limit of each method is $T = 60 \cdot n$ seconds. If an algorithm stops before T , it is restarted and the best solution is returned to the user.

GR starts from an empty solution and selects the next job to schedule with the largest gain g_j (ties are broken randomly). A_j denotes the set of feasible time slots for job j (i.e., not used by any job incompatible with j). If $p_j - |A_j| > K - C(s)$, job j is rejected. Otherwise, p_j slots are sequentially assigned to j and two situations can occur at each step: (1) if $p_j - |A_j| < 0$, the slot minimizing f_2 is chosen; (2) if $p_j - |A_j| \leq K - C(s)$, the slot minimizing the number of additional conflicts is selected (but j is rejected if more than K conflicts are created).

In *DLS*, a move consists in rescheduling a job j . The way to reassign p_j slots to j depends on A_j . If $A_j \geq p_j$, p_j slots are sequentially chosen in A_j while minimizing f_2 . Otherwise, the p_j slots are given one by one, by assigning at each step the slot minimizing the number of additional conflicts. Then, to maintain feasibility, some conflicts are removed with the following *Repair* method: while $C(s) > K$, the job involved in the largest number of conflicts is rejected (break ties with the gains). In *TS*, when a job j is rescheduled, it cannot be rescheduled for $tab = 10$ iterations. In *TS^R*, *TS* is restarted every $I = 100$ iterations.

DRM [27] relies on a pool of solutions on which tabu search works in turn. At each *generation*, a solution of the pool is first deconstructed, then reconstructed, and finally improved. A pool *Pop* of 10 solutions is handled. It is initialized by

generating 10 random solutions as follows. First, all the jobs of J are scheduled randomly, then, feasibility is reestablished with *Repair*, and finally the solution is improved with *TS* during $I = 100$ iterations. *DRM* uses a deconstruction parameter q which is initially set to $q_{min} = n/20$ and cannot exceed $q_{max} = n/3$. These two parameters were tuned based on the following ideas for controlling the diversification ability of the overall method. On the one hand, if q_{min} is too small, the method will not be able to escape from the current zone of the solution space. On the other hand, if q_{max} is too large, the deconstruction process will be similar to a restart. The pseudo-code of *DRM* is presented in Algorithm 2, where s^b and s^w respectively denotes the best and worst solution of *Pop*. On the one hand, *DRM* uses elements of *strategic oscillation* methods (see steps (2) and (3)): it explores unfeasible solutions but the distance from the feasibility border is controlled, as $K + q$ conflicts are allowed. On the other hand, *DRM* has features from *variable neighborhood search* (see steps (2) and (7)): it generates a deconstructed solution at a certain distance q from s , and q is updated according to the improvement or not of the best encountered solution.

Algorithm 2 *DRM*: Deconstruction-reconstruction algorithm

While T is not reached, do

1. Select the least frequently chosen solution s in the population *Pop*.
 2. *Deconstruction*: reject q jobs in s , chosen randomly.
 3. *Reconstruction*: schedule some jobs (chosen randomly) until $C(s) = q + K$. The slots are assigned one by one to each job, while minimizing the number of conflicts (break ties with f_2). If ties occur again, they are broken with information from *Pop*: the slot t maximizing $\sum_{i \in J_t} Sim(i, j)$ is chosen, where J_t is the set of jobs processed during slot t , and $Sim(i, j)$ is the number of slots where jobs i and j are performed simultaneously in the solutions of *Pop*.
 4. *Reestablish feasibility*: while s has above K conflicts, reject the job j with the smallest ratio $g_j/C_j(s)$, where $C_j(s)$ is the number of conflicts involving j in s .
 5. *Local search*: apply *TS* during I iterations, and denote s' the resulting solution.
 6. *Update Pop*: if s' is better than s^w , replace s^w with s' in *Pop*.
 7. *Update q* : if s' is better than s^b , set $q = q_{min}$; otherwise set $q = 1.05 \cdot q$ (if allowed).
-

4.3.3 Experiments

An instance (n, τ) is defined by its number n of jobs and its rate τ of allowed conflicts, from which it is deduced that $K = \tau \cdot n$. 15 instances were generated, with $n \in \{50, 100, 200\}$ and $\tau \in \{0, 0.02, 0.04, 0.1, 0.2\}$. Two jobs are incompatible with probability 0.5. Each p_j is randomly chosen in interval $[1, 10]$. The gain g_j is related to p_j as follows: a random number β is first chosen in interval $[1, 20]$, and $g_j = \beta \cdot p_j$ is set. Finally, the deadline D was set small enough to prevent the scheduling of all jobs. The algorithms were implemented in C++ and executed on a computer with a processor Intel Quad-core i7 2.93 GHz with 8 GB of DDR3 RAM memory.

Ten runs per instance were performed with $T = 60 \cdot n$ seconds. The results are given in Table 4.2, which shows for each method the average percentage gap according to the best ever found value for each objective (f_1, f_2). *TS* outperforms *GR*, which is slightly better than *DLS*: the obtained f_1 gaps are respectively 5.46%, 8.68% and 9.32%. The deconstruction and reconstruction steps in *DRM* are efficient, as the *DRM* gap is 2.29% for f_1 versus 6.87% for TS^R . *DRM* obtained the best results on 13 instances. Note that the smaller is the f_1 gap, the larger is the f_2 gap, as f_1 and f_2 are conflicting objectives.

Table 4.2 Results

n	K	Greedy	<i>DLS</i>	<i>TS</i>	TS^R	<i>DRM</i>
50	0	(4.21, 0)	(3.65, 0)	(4.62, 0)	(3.29, 8)	(2.1, 4)
50	1	(6.74, 0)	(6.41, 4)	(5.65, 20)	(4.7, 20)	(1.17, 20)
50	2	(8.33, 0)	(8.61, 0)	(3.78, 0)	(5.35, 0)	(0.82, 3.33)
50	5	(5.63, 0)	(5.97, 0)	(4.62, 0)	(2.86, 0)	(0.49, 0)
50	10	(2.17, 0)	(2.63, 10)	(2.38, 0)	(0.92, 0)	(0.33, 0)
100	0	(8.41, 16.67)	(9.26, 16.67)	(4.12, 13.33)	(8.04, 13.33)	(2.17, 20)
100	2	(8.06, 0)	(8.92, 10)	(6.95, 13.33)	(7.09, 16.67)	(3.93, 20)
100	4	(9.78, 0)	(10.68, 0)	(10.4, 0)	(7.99, 0)	(2.25, 0)
100	10	(10.84, 16.67)	(10.72, 13.33)	(8, 16.67)	(7.1, 20)	(2.56, 30)
100	20	(8.98, 0)	(9.52, 0)	(7.76, 8.57)	(5.95, 8.57)	(1.32, 17.14)
200	0	(7.88, 14.29)	(8.71, 11.43)	(0.32, 17.14)	(7.18, 14.29)	(5.09, 14.29)
200	4	(5.61, 0)	(6.24, 0)	(2.19, 2.5)	(4.51, 0)	(2, 0)
200	8	(7.69, 0)	(8.25, 0)	(3.92, 5)	(6.15, 0)	(0.97, 12.5)
200	20	(9.85, 2.5)	(10.45, 0)	(5.2, 7.5)	(7.53, 2.5)	(1.65, 12.5)
200	40	(9.65, 0)	(10.45, 0)	(5.76, 12.5)	(7.19, 5)	(0.96, 22.5)
Average		(8.68, 5.01)	(9.32, 5.14)	(5.46, 9.65)	(6.87, 8.04)	(2.29, 14.89)

4.4 Tabu Search with Diversity Control and Simulation

4.4.1 Presentation of the Problem

In most inventory management problems, two types of decision have to be taken at the manufacturer level: *when* and *how much* to order to suppliers [17]. It is assumed that setup, carrying and shortage costs are encountered during the year. Usually, inventory management models are characterized by stochastic demand and constant lead times. In contrast, this study, which generalizes the approach proposed in [16], deals with the situation where there is a constant known demand rate, but probabilistic lead times whose probability distributions change seasonally. Moreover, the lead times for different orders are assumed to be independent, thus crossovers can occur. Therefore, the interactive effects between different cycles (a cycle is defined

as the time between two consecutive orders) due to the occurrence of shortages are difficult to model. Consequently, even if the annual *approximated* costs can be analytically computed with a mathematical function f , simulation (of the lead times) is the only way to compute the annual *actual* costs F of a solution. Such a context is motivated by the management of raw material at a sawmill in North America. Without loss of generality, consider a 52-weeks planning horizon (a time period is a week). A solution (P, S) can be modeled by two vectors P and S defined as follows: $P_t = 1$ if an order occurs at the beginning of period t , and $P_t = 0$ otherwise; S_t is the order-up-to-level of available inventory at the beginning of period t if $P_t = 1$, and $S_t = 0$ if $P_t = 0$. The following reasonable assumptions are made: (A1) it is possible to analytically *approximate* the annual costs with a function $f(P, S)$ relying only on P, S and the probability distributions of the lead times; (A2) it is possible to compute the $F(P, S)$ (i.e., the annual *actual* costs) with a simulation tool; (A3) based on f , it is possible to analytically compute S from P with a so-called *Compute*($S | P$) procedure. It means that anytime P is modified, its associated S vector is immediately updated with *Compute*($S | P$).

4.4.2 Solution Methods

Due to the non-stationarity in the lead time distribution, the problem is combinatorial in nature (choice of the P_t 's and the S_t 's). Moreover, simulation is required to compute the actual cost of a solution. Thus, it makes sense to use (meta)heuristics. The solution space $X(N)$ is defined as the set of all the solutions (P, S) with $\sum_{t=1}^{52} P_t = N$. The general approach consists in providing good solutions for different solutions spaces, starting with $U(N)$ orders and ending with $L(N)$ orders, where $U(N) \leq 52$ (resp. $L(N) \geq 1$) is an upper (resp. a lower) bound on N . At the end, the best solution (over all the considered $X(N)$'s) is returned to the user.

For a fixed solution space $X(N)$, the following steps are performed: (S1) generate an initial solution (P, S) with N orders as equi-spaced as possible; (S2) based on f , try to reduce the approximate costs of (P, S) with a tabu search $TS_f(P, S)$ working on P ; (S3) based on F and without changing P , apply a descent local search $DLS_F(S | P)$ working on S (a move consists in augmenting or reducing one of the S_t 's, by one unit). In $TS_f(P, S)$, a move consists in putting an order earlier or later, but without changing the global sequence of orders. At each iteration, the best non-tabu move is performed. If an order is moved, then it is forbidden (tabu) to move it again for *tab* (parameter depending on N) iterations. The stopping condition is a maximum number *Iter* (parameter) of iterations without improvement of the best visited solution.

An extension of $TS_f(P, S)$, denoted $TS_f^M(P, S)$, is now proposed for step (S2). Instead of providing one solution, an idea is to provide a set M containing m (parameter) promising local optima (promising according to the *quality* function f and a *diversity* function $Div(M)$). To achieve this, additional ingredients are defined. The *distance* between P and P' is defined as $Dist(P, P') = \sum_{t=1}^{52} |P_t - P'_t|$. The

average *distance* between P and a set M of solutions is defined as $Dist(P, M) = \frac{1}{|M|} \sum_{P' \in M} Dist(P, P')$. The *diversity* of a set M of solution is computed as $Div(M) = \frac{1}{|M|} \sum_{P \in M} Dist(P, M - \{P\})$. M is initialized with solutions randomly generated. Let P be a solution found by tabu search at the end of an iteration. The key idea is the following: P should replace a bad (according to f) solution of M which poorly contributes to its diversity $Div(M)$. More precisely, let M' be the subset of M containing the m' (parameter) worst solutions of M , for which the worst value is f^{**} . Let $P^{(div)}$ be the solution of M' such that $P^{(div)} = \arg \min_{P' \in M'} Dist(P', M - \{P'\})$. Then, if $f(P) > f^{**}$, M is not updated. Otherwise, if $Dist(P^{(div)}, M - \{P^{(div)}\}) < Dist(P, M - \{P^{(div)}\})$, then P replaces $P^{(div)}$.

The resulting metaheuristic is summarized in Algorithm 3. The returned solution is (P^*, S^*) with an actual cost of F^* , which is the best solution visited in all the considered solution spaces.

Algorithm 3 General approach

Initialization: set $F^* = \infty$ and $N = UB(N)$

While $N \geq LB(N)$, **do**

1. generate an initial solution P with N orders as equi-spaced as possible
 2. apply $TS_f(P, S)$ or $TS_f^M(P, S)$, and let $M = \{P^{(1)}, \dots, P^{(m)}\}$ be the resulting set of local optima according to f ($m = 1$ if $TS_f(P, S)$ is used)
 3. for $i = 1$ to m , do: apply $DLS_F(S | P)$ on $(P^{(i)}, S^{(i)})$
 4. set $(P, S) = \arg \min_{i \in \{1, \dots, m\}} F(P^{(i)}, S^{(i)})$
 5. if $F(P, S) < F^*$, set $(P^*, S^*) = (P, S)$, and $F^* = F(P, S)$
 6. reduce N by one unit
-

4.4.3 Experiments

The experiments were performed on a PC Pentium 4 (1.6GHz/1 Go RAM). The parameters $Iter$, m and m' were respectively tuned to 1000, 10, 3. As the proposed method has to plan the orders for a whole year, the computing time is not an issue (but all the proposed methods never exceed an hour of computation). Each instance is characterized by its cost parameters (the fixed setup cost A per order, the inventory cost h per unit per period, the shortage cost B per missing unit). For each period t is known the minimum (resp. most likely and maximum) lead time a_t (resp. m_t and b_t). From these three values, discrete triangular distributions can be easily constructed. Two types T_1 and T_2 of instances were generated according to two sets of lead time distributions, with 24 instances per type (which differ according to A , h and B). Set T_1 is based on realistic data from the sawmill context, and is characterized by $a_t \in \{2, 5\}$, $m_t \in \{3, 7\}$, and $b_t \in \{6, 13\}$. Set T_2 , which represents a form of sensitivity analysis (the variation of the lead times is larger), is characterized by $a_t \in \{1, 8\}$,

$m_t \in \{2, 10\}$, and $b_t \in \{5, 16\}$. In Table 4.3 is provided a summary of the average percentage improvements (over a basic constructive heuristic based on an EOQ analysis) provided by the general proposed approach relying on $DLS_f(P, S)$ (where a descent local search is performed at step (S2) instead of tabu search), $TS_f(P, S)$ and $TS_f^M(P, S)$, respectively. Note that the percentage improvement computation is based on the difference between the reference cost (based on EOQ) and the considered provided cost, and this difference is then divided by the reference cost. The results are shown for three levels of B and for the two sets T_1 and T_2 . Unsurprisingly, the potential benefits of the three methods augments as the seasonality is increased. One can observe that $TS_f^M(P, S)$ outperforms $TS_f(P, S)$, and both methods are better than $DLS_f(P, S)$.

Table 4.3 Results on two sets of instances

Method	Set T_1			Set T_2		
	Small B	Average B	Large B	Small B	Average B	Large B
$DLS_f(P, S)$	1.39	1.52	1.61	3.75	3.58	3.47
$TS_f(P, S)$	1.72	1.82	2.01	4.15	4.05	3.74
$TS_f^M(P, S)$	1.82	1.86	2.16	4.18	4.06	3.79

4.5 Dynamic Tabu Search for a Resource Allocation Problem

4.5.1 Presentation of Dynamic Tabu Search

Let $X = (X^1, X^2, \dots, X^u)$ be a solution of a problem which consists in maximizing an objective function f . Each X^i is a vector of size $s(i)$ and can be denoted $X^i = (x_1^i, x_2^i, \dots, x_{s(i)}^i)$, where the x_j^i 's are real number. The following limitation constraint has to be satisfied for each i : $\sum_j x_j^i = c^i$. As random events can occur, it is assumed that the objective function f can only be evaluated with a simulation tool. In such a context, within a local search framework, it is straightforward to define a move in three steps: (A) select a decision variable type i ; (B) augment (resp. reduce) an x_k^i by an amount of w ; (C) reduce (resp. augment) some other x_j^i 's (with $j \neq k$) by a total amount of w (in order to satisfy the limitation constraint). Within a tabu search framework, if a decision variable x_k^i is augmented (resp. reduced), it is then forbidden to reduce (resp. augment) it during $tabu^i$ (parameter) iterations. The three key issues are now: (I1) which type of decision variable should be selected in (A); (I2) what is the amplitude w of the move in (B); (I3) how should the solution be adjusted in (C).

According to issue (I1), it is proposed to consider u types of *phase* in the solution method: each phase of type i works on X^i without modifying the other X^l 's ($l \neq i$). Each phase of type i can be performed during I_{max}^i (parameter) uses of the simulator. Working with phases (i.e., on one decision variable type at a time) allows to have a better control on the search.

To tackle issue (I2), it is proposed to dynamically update the move amplitude w during each phase of the search, within interval $[w_{min}^i, w_{max}^i]$ (parameters). Each phase starts with $w = w_{max}^i$, and anytime I^i (parameter) iterations without improvement of the best encountered solution X^* have been performed, w is reduced by δ^i (parameter), but never under w_{min}^i . If a move leads to a solution better than X^* , the process restarts with $w = w_{max}^i$, and so on. This strategy allows to progressively focus on a promising region of the solution space. Such a technique has common points with variable neighborhood search [11].

Issue (I3) depends on the two other issues: if x_k^i has been selected for a variation of w , the search process should focus on that decision and not modify as much the other decision variables (of the same type) in order to adjust the solution with respect to the associated constraint c^i . Thus, if x_k^i was augmented (resp. reduced) by w , the process should then equally reduce (resp. augment) the $s(i) - 1$ other variables (of the same type) by a total amount of w , which means that each decision variable is in average reduced (resp. augmented) by $\frac{s(i)-1}{w}$. The resulting DTS method (for *Dynamic Tabu Search*) is summarized in Algorithm 4, which returns the best encountered solution X^* with value f^* . At each iteration, the neighbor solution can be the best among a set of N (parameter) candidate neighbor solutions.

Algorithm 4 DTS: Dynamic tabu search

Initialization

1. Generate an initial solution $X = (X^1, X^2, \dots, X^u)$.
2. Initialize the best encountered solution: set $X^* = X$ and $f^* = f(X)$.
3. Set $i = 1$ and $w = w_{max}^i$.

While the simulation software has not been used q (parameter) times, do

1. generate a non-tabu neighbor solution \hat{X}^i of X^i by modifying a decision variable x_k^i of X^i by w ;
 2. update the current solution: set $X^i = \hat{X}^i$;
 3. update the move amplitude w :
 - if I^i iterations without improving X^* have been performed, set $w = w - \delta^i$;
 - if $w < w_{min}^i$, set $w = w_{min}^i$;
 - if $f(X) > f^*$, set $w = w_{max}^i$;
 4. update the best encountered solution: if $f(X) > f^*$, set $X^* = X$ and $f^* = f(X)$;
 5. update the tabu tenures: it is forbidden to modify x_k^i in the reverse way for tab^i iterations;
 6. next phase: if I_{max}^i runs of the simulator have been performed, set $i = (i \bmod u) + 1$ and $w = w_{max}^i$;
-

4.5.2 Application to a Resource Allocation Problem

DTS is an appropriate solution method to dimension assembly/disassembly production systems. By dimensioning, one can refer to maximizing the production rate of a machine without successors, with respect to limited resources (e.g., buffer capacity between the machines, total cycle time of the machines). Relevant recent papers in the field are [5, 15]. As random failures might occur on the machines, a professional software is used to evaluate a solution [2].

Consider a production system with m machines and n buffer zones, modeled by a graph $G = (V, A)$ with vertex set V and arc set A . Vertex v represents machine v and there is an arc (v, v') from v to v' if a piece processed on machine v has then to be processed on machine v' . Moreover, each arc (v, v') also represents a buffer (i.e., a limited zone where are stored the pieces between the associated machines). Two types of decision variables (i.e., resource types) are considered: the designed cycle time t_v for machine v , and the buffer capacity $b_{vv'}$ allocated to arc (v, v') . A solution for a production network with $m = 9$ and $n = 8$ is presented in Fig. 4.1. The cycle time associated with machine 1 is $t_1 = 8$, and the buffer capacity between machines 1 and 3 is $b_{13} = 43$. The considered limitations are 60 for the total cycle time (i.e., $\sum_{v \in V} t_v = 60$) and 320 for the buffer capacity (i.e., $\sum_{(v, v') \in A} b_{vv'} = 320$).

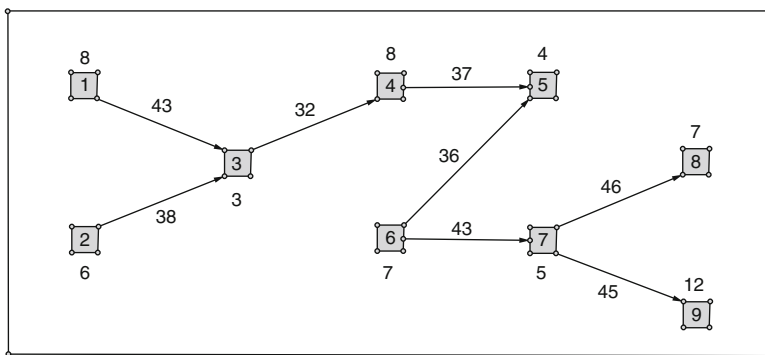


Fig. 4.1 Graph representation of a production system

The DTS approach showed a very good performance on such a production system [26]. Indeed, it was tested on the production network associated with Fig. 4.1, for which each machine has its own role: machines 1, 2, 4, 8 and 9 are classical processing machines, machines 3 and 5 are assembling machines, and machines 6 and 7 are disassembling machines. The objective consists in maximizing the production rate of machine 5. The breakdown probability is 5% (associated with each time step) and its length is generated with a uniform distribution in interval [100, 800]. DTS was compared with a descent local search DLS (the same algorithm as DTS, but without considering tabu tenures), and a classical tabu search TS for

which at each iteration, a move consists in augmenting/reducing any decision variable by any possible amount (followed by the adjustment of the other variables of the same type in order to meet the upper bounds). In TS, a sample of all possible amplitudes is considered at each iteration to modify the decision variable. A pool of 50 initial solutions were generated, which have an average production rate of 1.31 (pieces/minute). DLS was able to reach an average production rate of 1.39, TS obtained 1.37, and DTS reached 1.41. It was also observed that the quality of the resulting solution negligibly depends on the initial solution, which indicates that DTS is a robust approach.

4.6 Conclusion

In this chapter, enhanced tabu search approaches are discussed for four domains: car sequencing, job scheduling, resource allocation and inventory management. Because of the specificities of each problem, classical tabu search procedure are likely to be inefficient if its intensification and diversification abilities are not appropriately handled. For this purpose, the success of three mechanisms is discussed, namely the controlled use of various neighborhood structures, the tactical management of restarts, and a strategic deconstruction and reconstruction technique. Such ingredients allows to design *all-terrain* tabu search metaheuristics, as it results in a good balance between exploitation and exploration, allowing an efficient control on the search process. Finally, we would like to mention that other *all-terrain* approaches were also successfully adapted in other fields (e.g., [9, 10, 14, 22]).

References

1. C. Almeder, B. Almada-Lobo, Synchronisation of scarce resources for a parallel machine lotsizing problem. *Int. J. Prod. Res.* **49**(24), 7315–7335 (2011)
2. Rockwell Automatisation, Arena Standard Edition, Milwaukee (2000)
3. C. Becker, A. Scholl, A survey on problems and methods in generalized assembly line balancing. *Eur. J. Oper. Res.* **168**(3), 694–715 (2006)
4. I. Bloechliger, N. Zufferey, Multi-coloring and project-scheduling with incompatibility and assignment costs. *Ann. Oper. Res.* **211**(1), 83–101 (2013)
5. A. Dolgui, A. Ereemeev, V. Sigaev, HBBA: hybrid algorithm for buffer allocation in tandem production lines. *J. Intell. Manuf.* **18**(3), 411–420 (2007)
6. G. Even, M.M. Halldórsson, L. Kaplan, D. Ron, Scheduling with conflicts: online and offline algorithms. *J. Sched.* **12**(2), 199–224 (2009)
7. M. Garey, D.S. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979)
8. M. Gendreau, J.-Y. Potvin. *Handbook of Metaheuristics*. International Series in Operations Research and Management Science, vol. 146 (Springer, Berlin, 2010)
9. A. Hertz, D. Schindl, N. Zufferey, Lower bounding and tabu search procedures for the frequency assignment problem with polarization constraints. *4OR* **3**(2), 139–161 (2005)

10. A. Hertz, D. Schindl, N. Zufferey, A solution method for a car fleet management problem with maintenance constraints. *J. Heuristics* **15**(5), 425–450 (2009)
11. N. Mladenovic, P. Hansen, Variable neighborhood search. *Comp. Oper. Res.* **24**, 1097–1100 (1997)
12. J. Respen, N. Zufferey, Tabu search with guided restarts for a car production problem with a 2/3 balancing penalty, in *Proceedings of the 5th International Conference on Metaheuristics and Nature Inspired Computing*, Marrakech, 27–31 October 2014
13. J. Respen, N. Zufferey, E. Amaldi, Heuristics for a multi-machine multi-objective job scheduling problem with smoothing costs, in *1st IEEE International Conference on Logistics Operations Management*, vol. LOM 2012, Le Havre, 17–19 October 2012
14. D. Schindl, N. Zufferey, Solution methods for fuel supply of trains. *Inf. Syst. Oper. Res.* **51**(1), 22–29 (2013)
15. C. Shi, S.B. Gershwin, An efficient buffer design algorithm for production line profit maximization. *Int. J. Prod. Econ.* **122**(2), 725–740 (2009)
16. E.A. Silver, N. Zufferey, Inventory control of raw materials under stochastic and seasonal lead times. *Int. J. Prod. Res.* **43**, 5161–5179 (2005)
17. E.A. Silver, D.F. Pyke, R. Peterson, *Inventory Management and Production Planning and Scheduling* (Wiley, New York, 1998)
18. S.A. Slotnick, Order acceptance and scheduling: a taxonomy and review. *Eur. J. Oper. Res.* **212**(1), 1–11 (2011)
19. C. Solnon, V.D. Cung, A. Nguyen, C. Artigues, The car sequencing problem: overview of state-of-the-art methods and industrial case-study of the ROADEF’ 2005 challenge problem. *Eur. J. Oper. Res.* **191**(3), 912–927 (2008)
20. S. Thevenin, N. Zufferey, A deconstruction-reconstruction metaheuristic for a job scheduling problem, in *Proceedings of the 5th International Conference on Metaheuristics and Nature Inspired Computing*, Marrakech, 27–31 October 2014
21. S. Thevenin, N. Zufferey, J.-Y. Potvin, Multi-objective parallel machine scheduling with incompatible jobs, in *Proceedings for the ROADEF 2014 Conference*, Bordeaux, 26–28 February 2014
22. S. Thevenin, N. Zufferey, M. Widmer, Metaheuristics for a scheduling problem with rejection and tardiness penalties. *J. Sched.* **18**(1), 89–105 (2015)
23. N. Zufferey, Dynamic tabu search with simulation for a resource allocation problem within a production environment, in *Proceedings of 4th International Conference on Metaheuristics and Nature Inspired Computing (META 2012)*, Sousse (2012)
24. N. Zufferey, Metaheuristics: some principles for an efficient design. *Comput. Technol. Appl.* **3**(6), 446–462 (2012)
25. N. Zufferey, Tabu search with diversity control and simulation for an inventory management problem, in *Proceedings of the 5th International Conference on Metaheuristics and Nature Inspired Computing (META 2014)*, Marrakech, 2014
26. N. Zufferey, Multiple neighborhood in tabu search: successful applications in operations management, in *Proceedings of the 4th M-Sphere Conference*, Dubrovnik, October 2015
27. N. Zufferey, F. Glover, A reconstructive evolutionary metaheuristic for the vertex coloring problem, in *Proceedings of the 23rd Benelux Conference on Artificial Intelligence*, Gent, 3–4 November 2011, pp. 352–357
28. N. Zufferey, M. Vasquez, A generalized consistent neighborhood search for satellite scheduling problems. *RAIRO Oper. Res.* **41**(1), 99–121 (2015)

Chapter 5

A Re-characterization of Hyper-Heuristics

Jerry Swan, Patrick De Causmaecker, Simon Martin, and Ender Özcan

Abstract Hyper-heuristics are an optimization methodology which ‘search the space of heuristics’ rather than directly searching the space of the underlying candidate-solution representation. Hyper-heuristic search has traditionally been divided into two layers: a lower problem-domain layer (where domain-specific heuristics are applied) and an upper hyper-heuristic layer, where heuristics are selected or generated. The interface between the two layers is commonly termed the “domain barrier”. Historically this interface has been defined to be highly restrictive, in the belief that this is required for generality. We argue that this prevailing conception of domain barrier is so limiting as to defeat the original motivation for hyper-heuristics. We show how it is possible to make use of domain knowledge without loss of generality and describe generalized hyper-heuristics which can incorporate arbitrary domain knowledge.

Keywords Hyper-heuristics • Metaheuristics • Optimization • Machine learning • Constraint programming

J. Swan (✉)

Computer Science, University of York, York, UK

e-mail: jerry.swan@york.ac.uk

P. De Causmaecker

Computer Science, KU Leuven, Kortrijk, Belgium

e-mail: patrick.decausmaecker@kuleuven.be

S. Martin

Computer Science, University of Stirling, Stirling, UK

e-mail: spm@cs.stir.ac.uk

E. Özcan

School of Computer Science, University of Nottingham, Nottingham, UK

e-mail: ender.ozcan@nottingham.ac.uk

5.1 Introduction

Sörensen and Glover [38] define a *metaheuristic* as “a high-level problem independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms”. The goal of *hyper-heuristics* is to act as effective cross-domain search methodologies, i.e. to be applicable not only to instances with different characteristics from a single domain, but also across multiple problem domains. The definition of a hyper-heuristic varies considerably in the literature: indeed, interpreting the notion of ‘searching the space of heuristics’ in full generality allows application to *any* heuristically-informed solution mechanism (e.g. the choice of pivot function used in Quicksort [43]). In this article, we concentrate on the application of hyper-heuristics to metaheuristic search. In the following sections, we describe how the definition of hyper-heuristics has evolved over time. We re-visit the underlying motivation in order to highlight some popular misconceptions and the attendant need for re-characterization.

5.1.1 Historical Development of Hyper-Heuristics

One of the earliest studies in this area was an application to a job shop scheduling problem due to Fisher and Thompson [16]. The use of scheduling (dispatching) rules as heuristics is common in this area and the study was motivated by the idea of “a combination of the two rules being superior to either one separately”. In the early 1990s, Storer et al. [41, 42] proposed a general approach combining heuristic- and solution- space methods for solving sequencing problems. The authors used job shop scheduling as a case study and argued that the proposed approach can be “easily” applied to any scheduling objective. Fang et al. [14, 15] subsequently evolved sequences of heuristics for constructing schedules, explaining how the proposed approach can be “simply amended” to deal with more complex industrial open shop scheduling problems.

The term ‘hyperheuristics’ (in unhyphenated form) was first introduced by Cowling et al. [12] as a means of deciding which low level heuristic to apply during the search process, depending on the nature of the region being explored. This initial definition referred only to (what has become known as) ‘selective’ hyper-heuristics, with generative hyper-heuristics being a later development [34]. The motivation for the use of the term ‘hyper’ comes from hypergraphs, where an edge is an n -ary relation on vertices, the analogy being that hyper-heuristic selection is performed on a *collection* of operators (i.e. functions with signature $Op : S \rightarrow S$, for some candidate solution representation S). Hyper-heuristic selection thus takes a list of operators, together with a function for choosing an operator from this list and applies the selected operator to an incumbent state. Mathematically, we can represent this as:

$$\begin{aligned} \text{select} &: [Op] \times ([Op] \rightarrow Op) \times S \rightarrow S \\ \text{select} &: (\text{operators}, \text{choose}, \text{incumbent}) \mapsto \text{choose}(\text{operators})(\text{incumbent}) \end{aligned}$$

If, as has invariably been the case, the list of operators and the function for choosing from them are known in advance, then the signature for *select* can be considered to be $S \rightarrow S$, precisely that of an operator. This notion of ‘recursive composition via selection’ [49] could equally be applied to other metaheuristic components (i.e. acceptance, termination etc.), though the authors are not aware of any such approaches (e.g. the evolution of acceptance criteria by Hyde et al. [22] was obtained via a *generative* rather than selective approach).

Cowling et al. [12] stated that a hyper-heuristic approach operates at ‘a higher abstraction level’ than a metaheuristic and in practice this has been translated as ‘operating independently of the underlying problem domain’. To this end, Cowling et al. [12] introduced the notion of a *domain barrier* between the layers of hyper-heuristic framework and problem-domain implementation. As we explicitly demonstrate in Sect. 5.2, this notion of domain independence can be described purely in terms of *generic metaheuristics*, avoiding the rather mixed collection of concepts that has become associated with hyper-heuristics (see also Fig. 5.1).

The widely-cited definition due to Burke et al. [8] of “(meta-)heuristics to choose (meta-)heuristics” has the stated motivation of “raising the level of generality at which optimisation systems can operate”. The authors contemplate that many businesses, particularly small ones, are interested in “good enough, soon enough, cheap enough” solutions to their problems. Given the high cost of developing problem-specific methods, this highlights the need for a general, easy-to-use, yet robust approach for ‘providing near optimal solutions’. The intention was that the domain barrier represents the separation between different levels of expertise, i.e. practitioners would be responsible for implementing only solution representations and naïve ‘knowledge-poor’ (and hence presumably often randomized) heuristics for each new problem domain, with researchers tasked with devising hyper-heuristics which work well across domains.

Ross et al. [35, 36] defined a hyper-heuristic as a search method which combines simple heuristics to solve a range of problems satisfactorily. They evolved bin-packing rules using a Learning Classifier System [35] which learns which low level heuristic to use at a given decision point. Ross [34] provided a similar definition as Soubeyga [39] introducing hyper-heuristics as “heuristics to choose heuristics” and combining multiple heuristics to compensate for individual weaknesses. In a foundational paper on generative hyper-heuristics, Ross [34] further proposed hyper-heuristics as a special form of genetic programming, with a function set consisting of existing heuristics. In this study, the aim in hyper-heuristic design is presented as finding a “fast, reasonably comprehensible” approach, repeatedly able to produce high quality solutions. Qu et al. [31] proposed a tabu search approach to examination timetabling and graph colouring problems, indirectly acting on the candidate solutions via a mixture of graph-colouring heuristics. Two cross-domain heuristic search competitions, CHeSC 2011 and 2014 were performed using the HyFlex selective hyper-heuristics framework [28], which provided an implementation of six problem domains.

A recent definition of hyper-heuristic which is probably the most commonly-used is provided by Burke et al. [10] as “a search method or learning mechanism for selecting or generating heuristics to solve computational search problems”. A more concrete definition adopts the terminology of the ‘Algorithm Selection Problem’ (ASP) [33] to describe hyper-heuristics as ‘a mapping from features to algorithms’. A rich research area that has historically been more overtly influenced by the ASP than hyper-heuristics is the field of algorithm portfolios [19, 21]. Recent work in this field includes ‘Dynamic Algorithm Portfolios’ [17] which chooses from a subset of available algorithms, applying them simultaneously to a problem instance until the fastest algorithm solves it.

In principle, the adoption of the ASP would allow the gamut of machine learning techniques to be applied to hyper-heuristics, but in practice the features made available for learning by selective hyper-heuristics have been limited. In contrast, the input to *generative* hyper-heuristics is (necessarily) domain-specific, and the only general framework supporting generative hyper-heuristics which we are aware of is TEMPLAR [43]. Chakhlevitch and Cowling [11] specifically argue the importance of limited problem domain information in achieving cross-domain generality for selective hyper-heuristics. Moreover, they further state that a hyper-heuristic would ideally be informed only of the number of low level heuristics for a given problem domain and objective value of a given solution. A variant of the strict notion of domain barrier due to Woodward et al. [48] has been perpetuated via HyFlex as a de facto standard.

As can be seen from the above, the definition of hyper-heuristic has evolved considerably over time. As a result, there is relatively little clear consensus on what the essential mechanisms of a hyper-heuristic actually are. Figure 5.1 is a feature diagram of various concepts historically associated with (selective) hyper-heuristics. The concepts which are non-obvious (or otherwise not covered above) are:

- Heterogeneous operators: The ability to treat different operators in a uniform manner in the hyper-heuristic layer. For example, with a permutation representation, the ability to mix e.g. 2-opt with transpositions.
- Selection a posteriori versus a priori: whether or not an operator must be applied (to the current incumbent solution) before it can be chosen. Metaheuristics are traditionally, ‘apply then choose’, e.g. choose the first- or best-improving. The a priori case is when an operator is chosen via some mapping based on its features and the search trajectory.

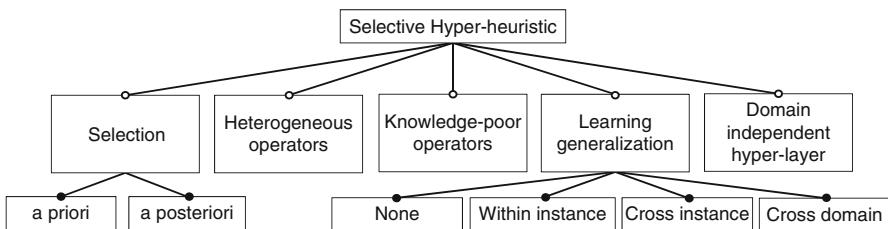


Fig. 5.1 Concepts historically associated with selective hyper-heuristics

Despite the diversity of concepts associated with hyper-heuristics, the authors are aware of only a few attempts to consolidate them. It is also interesting to note that some conceptual and formal approaches that might reasonably be included under the wider notion of ‘heuristics to select or generate heuristics’ (e.g. [24, 27]) have not historically been considered to be part of the literature. As discussed above, selective hyper-heuristics can be shown to be an instance of the well-known ‘Composite’ design pattern [49], a mechanism used by the HYPERION framework [44, 7] to allow the same source code to express both metaheuristics and hyper-heuristics. The widely-cited classification scheme due to Burke et al. [9] is generalized by Swan et al. [46] to allow any combination of selective/generative and online/offline to co-exist and interoperate at runtime within the same architecture.

5.1.2 Effectiveness in New Domains

It has been observed that not only the design of a selective hyper-heuristic but also the choice of predefined low level heuristics influences its performance [29]. To the best of the authors’ knowledge, there are no applications of selective hyper-heuristic for which the use of only ‘knowledge poor’ low-level heuristics is competitive with the state-of-the-art. In practice, state-of-the-art low level heuristics have therefore made their way into the domain implementations for improved performance, e.g. in several of the problem domains implemented by HyFlex [28].

This indicates that (selective) hyper-heuristic research has become disconnected from the original motivation, failing to provide solutions which are ‘good enough, cheap enough’ (and in general certainly not the ‘near optimal’ solutions which were originally hoped for). Due to the artificially-restricted notion of the domain barrier, devising and using selective hyper-heuristics is currently no less labour-intensive than simply using some generic metaheuristic framework (detailed reasons for this are given in Sect. 5.2). For researchers, it surely is clear that the application of machine learning (e.g. [4]) is necessary to avoid cross-domain generalization being obtained via laborious manual ‘generate and test’. However, the de facto domain barrier restrictions mean that even the elaborate machine learning techniques that have been employed in selective hyper-heuristics tend to make use of limited information. There is therefore a need to move from this restrictive interface to one which:

- Enables more expressive (i.e. feature-rich) hyper-heuristics.
- Allows state-of-the art knowledge to be easily incorporated into a new problem domain model by less-experienced practitioners.

To achieve this, it is necessary to disentangle approaches which have become prevalent from the goals they sought to achieve. This is particularly important since none of the concepts of Fig. 5.1 suffice to fully-characterize the many publications with ‘hyper-heuristic’ in the title. To reiterate: applying hyper-heuristics would be of interest to practitioners (e.g. in industry) if this avoids the need for labour-

intensive modelling of a specific problem domain. However, metaheuristic search is *already* a computational intelligence success story in this respect: approaches such as simulated annealing, tabu search, genetic algorithms and swarm optimization yield good (and often state-of-the-art) results in a wide range of problem domains. The minimal requirement for domain modelling using these techniques is very small, needing only a choice of solution representation, solution quality measure and one or more operators for perturbing/recombining solutions.

Depending on the available modelling budget, the sophistication of domain knowledge can range from the very naïve (e.g. potentially infeasible solutions; randomized operators, quality measure that does not yield a search gradient) through to a highly-informed combination of state-of-the-art techniques. The important point relative to selective hyper-heuristics is that, in this case, practitioners retain the option to increase the competence of the framework layer as required. In the next section, we show how the popular conception of selective hyper-heuristics can be viewed as a (somewhat uninformed) special case of *generic metaheuristics*.

5.2 Popular Notion of the Domain Barrier

The de facto conception of selective hyper-heuristics (e.g. as exemplified by Hyflex [28]) is shown in Fig. 5.2. Here the notion of ‘heuristic’ is restricted to that of ‘operator’, i.e. a perturbation of a candidate solution. Hyflex operates as follows: the hyper-heuristic solver maintains a list of heuristics $[o_1, \dots, o_n]$ and a list of solutions $[s_1, \dots, s_m]$. The heuristic value of a solution s_k is given by $e(s_k)$. At each iteration,

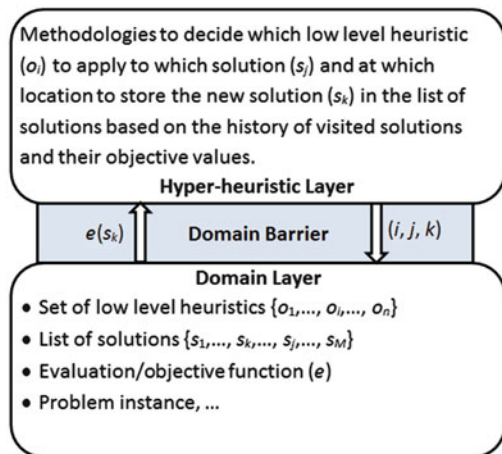


Fig. 5.2 A popular conception of selective hyper-heuristics

the solver chooses three integers i, j, k such that the solution in slot k is replaced by the result of applying operator i to solution j . This is the characterisation given by Woodward et al. [48] as:

$$Sel : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R} \quad (5.1)$$

$$Sel : (i, j, k) \mapsto e(o_i(s_k)) \quad (5.2)$$

with solution k having first being replaced by $o_i(s_j)$ as a side-effect. The only information available to the solver in making this choice is the fitness value/execution time resulting from operator application (and any memorization of such information from previous iterations). Problem-specific information is hidden in the belief that the generality of the hyper-heuristic will be lost. This can be considered as a ‘lowest-common-denominator’ approach to generality.

It should be immediately clear that this formulation is too restrictive to allow many popular metaheuristics to operate hyper-heuristically (not least) because of the following common requirements:

1. The ability to compare solutions for domain-specific equality. This arises since solution representations are only visible to the hyper-layer as integer indices. The result is that even such elementary techniques as ‘breadth-first search’ cannot be expressed.
2. The ability to determine those parts of a solution that have been changed by a heuristic. This is a common requirement in tabu-style approaches (e.g. making recently-perturbed permutation indices tabu in the TSP).
3. The ability to detect and react to constraint violations (e.g. infeasible solutions) at the framework level.

The inability to test for equality also precludes approaches such as reactive tabu or breakout local search [5, 6], which explicitly maintain equality-based solution histograms in order to determine when a diversification strategy should be triggered.

For clarity, we now make explicit the difference between metaheuristics and selective hyper-heuristics (as defined by the ‘traditional’ Hyflex-style domain barrier). Consider a generic framework for a local search metaheuristic shown in Listing 5.1. The framework is parameterized by the type S of solution representation and the type F denoting the features to be memorized in the search history. $[T]$ denotes a list of elements of type T and operators Op are functions $S \rightarrow S$. The memorized features are used for decision-making during the selection process and are obtained via a mapping $features : Op \times S \times S \rightarrow F$.

To instantiate this framework as a metaheuristic (e.g. simulated annealing for the TSP), we put:

- S as a permutation.
- $[Op]$ as any desired perturbations e.g. transpositions, 2-opt etc.
- $selectOp$ as uniform random selection.
- $accept$ as Metropolis-Hastings.
- Feature type F to be the empty set (in this particular case).

```

S search(incumbent: S, operators: [Op], history: [F]) {
    while(not finished(incumbent, history)) {
        Op op = selectOp(incumbent, operators, history);
        S incoming = op(incumbent);
        incumbent = accept(incumbent, incoming)
        history.update( features(op, incumbent, incoming) );
    }
    return incumbent;
}

```

Listing 5.1 Generic local search meta- or hyper-heuristic

To instantiate this framework as a Hyflex-style hyper-heuristic, we put:

- S as an integer in the range $[0, m)$.
- Op as an integer in the range $[0, n)$.
- $selectOp$ as e.g. choice function [23].
- Feature type F to be (Op, S, S, \mathbb{R}) , given by $(i, j, k, e(s_k))$ from Eq. (5.1), above.

It should be clear that the same arguments as given above apply to any other meta-heuristic, population-based or otherwise. Hyflex-style selective hyper-heuristics can therefore be seen as a special case of a metaheuristic in which solutions and operators are mapped onto opaque integer indices at the framework level. It should be clear from the fact that the framework level is *already generic in terms of solution representation*, that there is absolutely no requirement for this degree of opacity: we can instantiate the hyper-heuristic framework with arbitrary solution and operator types, thereby eliminating the above issues (equality of states etc.) associated with the opaque handles of Hyflex. In addition, such types can provide much greater utility without loss of generality, e.g. the ability to decompose a solution into parts to act as finer-grained tabu attributes.

5.3 The Need for ‘Domain-Independent Domain Knowledge’

As explained in the previous section, if one adopts the prevailing notion of the domain barrier, then the minimal responsibilities of a practitioner in implementing some new domain are precisely the same *irrespective of whether they wish to use metaheuristics or hyper-heuristics*. In fact, they are considerably worse off with the latter approach: if search quality is unsatisfactory, then there is no means of ‘injecting further domain knowledge’ at the framework level as can be done with a metaheuristic. What is therefore needed is a hyper-heuristic approach which can operate on much richer domain knowledge.

To illustrate the extent to which this is possible, it is useful to consider the distinction between ‘analytic’ and ‘empirical’ knowledge. We define the former to be information which is given a priori (or otherwise formally derived from) the prob-

lem description and the latter to be that derived from the solution trajectory. The de facto conception of hyper-heuristics is that they can only make use of empirical knowledge at the framework level. The empirical features exposed by HYFLEX are:

- Objective value arising from applying $o_i(s_k)$.
- Execution time for operator application.
- Integer handle of an operator.

Given this limited set of features, it is difficult to learn useful information even *within a domain*. Handles denoting operators have no persistent meaning across problem domains and the possibilities for cross-domain learning are therefore even more limited. While it is encouraging to see that extensions to the de facto conception of the domain barrier have recently been proposed [30], it is possible to go much further than this, as we now demonstrate.

A recent machine learning approach of Asta and Özcan [4] (in which linkage between operators is estimated via tensor factorization) is perhaps representative of the limits of what might be learned from the kind of empirical information described above. In contrast, in many cases we do not need to mine information of this form from the solution trajectory: we already have it as prior knowledge. Consider the ability to reason algebraically about operators. For example, it is well known that transpositions of permutations are self-inverse, so it is wasteful to apply the same transposition in succession. This is of course an extremely simple example: there is no limit to the kinds of exploitable analytic information we might devise. As a more general example, *any* sequence of operators (of any sort, as long as they have signature $S \rightarrow S$) forms an algebraic structure known as a *monoid* under concatenation. This monoid structure can be represented by equations between operators that describe which sequences always lead back to their starting state (irrespective of the specific start state). As shown by Swan et al. [45], it is often possible to use this algebraic relationship between operators to derive a set of *rewrite rules*. These rewrite rules allow any sequence of operators to be reduced a priori to its minimal-length equivalent, thereby eliminating redundancy (i.e. cycles) in the state-space graph. In particular, this is an example of cross-domain analytic knowledge: Swan et al. apply this to the Quadratic Assignment Problem, but the equational description of the associated monoid structure could be used in any problem which operates on permutations.

While this cannot be achieved hyper-heuristically with a HYFLEX-style formulation, making the additional information (in this case the monoid equations) available to a hyper-heuristic solver has absolutely no cost in terms of generality: a solver which is incapable of acting on such information can simply ignore it. The challenge is therefore to exploit such information without loss of generality as ‘domain-independent domain knowledge’. Although the notion of being able to operate on arbitrary features has always been implicit in the ASP,¹ the vast majority of work in selective hyper-heuristics has been concerned with the limited feature set described above.

¹ And to a less overt degree implied in early work on hyper-heuristics (e.g. [8]).

The need to move away from such unnecessary restrictions then leads immediately to considerations of knowledge-representation, which have fortunately been well-studied in the wider AI community for many years. Two examples of existing hyper-heuristic systems which can (in terms of their architectural principles) incorporate arbitrary domain knowledge are the blackboard system of Swan et al. [46] and the multi-agent system of Martin et al. [25]. Although aspects of their architectures differ, they both have the ability to associate competent, representation-aware algorithms (known as ‘knowledge sources’ or ‘agents’, respectively) with heterogeneous sources of information, and it is this which allows these frameworks to operate across domains. The actual association process (which is precisely a mapping from features to algorithms) might range from the relatively trivial (e.g. a dictionary of key-value pairs indicating that a particular algorithm is competent to operate on permutation representations) to more specialized condition-action patterns induced from any source of information (analytic or empirical) that the agent is able to recognize. In the next section, we discuss the use of constraint satisfaction as a generic vocabulary for expressing domain-independent domain knowledge.

5.4 Cross-Domain Knowledge Representation

We now elaborate on the desired nature of knowledge representation for use in a hyper-heuristic framework capable of generalized cross-domain learning. Such a generalized representation should ideally allow for the expression of problem specifications and the description of low-level heuristics, together with formal properties of those problems and heuristics. What we therefore require is a description in a problem-independent vocabulary: such a representation would explicitly support cross domain learning. We may then rely on analytic knowledge of widely-used problem representations such as graphs and tensors, or reductions to well-known problems. In principle the hyper-heuristic could access such a specification in any detail. In this respect, a hyper-heuristic need not differ from a metaheuristic specialized to some problem domain. As discussed above, the main goal of hyper-heuristic research is to act effectively in a newly-specified domain without relying on the presence of optimization experts. Analytic knowledge allows us to better achieve this by injecting richer domain knowledge into the search process. In particular, it can contain rules about operator applications which are known a priori to result in improvement. An well-known example is the uncrossing of edges in the TSP:

detect : *findcross*

\exists segment (A_p, A_{p+1}) and (A_{p+i}, A_{p+i+1}) in cycle $\{A_i | i = 1 \dots n\}$

 s.t. $|A_p A_{p+1}| + |A_{p+i} A_{p+i+1}| > |A_p A_{p+i}| + |A_{p+1} A_{p+i+1}|$

action : *uncross*

 replace $(A_p, A_{p+1}) \rightarrow (A_p, A_{p+i}), (A_{p+i}, A_{p+i+1}) \rightarrow (A_{p+1}, A_{p+i+1})$

 and reverse $\{A_{p+1} \dots A_{p+i}\}$.

Such representations could also contain statements about certain relationships or constraints which are (nearly) always satisfied. A domain expert may wish to express his experience that certain patterns never contribute to good solutions. A suitable knowledge representation formalism will permit efficient handling of such expressions.

One possible means of expressing cross-domain knowledge is constraint programming, which provides a well-established means for describing such generalised problems. Constraint satisfaction problems have the advantage of being declarative, i.e. allow the statement of constraints in an implementation-independent manner. Specifying patterns (in various forms) is common to many constraint languages and properties of low level operators and the relationships between them can readily be formulated in this manner. The detailed description of the workings of those operators may still proceed in any language. This implementation may moreover differ from one domain to another. Given an ontology of domain-independent concepts, it is very natural to express the transferable knowledge in terms of constraints.

To make these issues concrete, we proceed to discuss the features of a specific knowledge representation format. XCSP [2, 37, 50] allows expression of constraint satisfaction problems (CSP), weighted constraint satisfaction problems (WCSP) and quantified constraint satisfaction problems (QCSP). It allows the description of instances according to characteristics such as real-world; patterned; random instances with/without a structure or involving only boolean variables. The authors furthermore distinguish instances based on whether they are defining all constraints in extension, partly in intension or use global constraints. Designed for expression of general decision and optimization problems, it supports two notations: a full XML notation compromising between human and machine readability and an abridged notation which is much more human readable. Both notations are equivalent and translation in both directions is possible.

The XCSP specification of a problem domain for a hyper-heuristic can therefore contain information about problem properties and low level heuristics. Information on low level heuristics in present hyper-heuristic frameworks include categorizations such as ‘hill climber’ and ‘population based’ [30]. As argued elsewhere in this paper, it is possible to extend this much further: since XCSP allows for the introduction of arbitrary problem and data descriptions, then in principle any information about a low level heuristic that has been acquired should be expressible, in principle to any detail. For example, current hyper-heuristic practice requires that determining when it might be appropriate to call one heuristic immediately following another is achieved empirically. In many cases, this information is already available analytically, e.g. in the manner which is exploited in the ‘Reverse Elimination Method’ of tabu search [18].

Presently, tools are available for solving, parsing, checking of instances and solutions and shuffling variables (to check robustness of solvers). Since the format is open and has systematic parsers available, other tools may be conceived: one might think of discovering patterns in instances or history, item set mining and constraint learning, which could serve to summarize XCSP descriptions and keep them fit for use by on-line hyper-heuristics. Links between data mining and constraint

programming have been suggested before [13], and ongoing integration between the two could support a fundamental move from pure model-based problem solving algorithms to an integrated, data-driven approach. In this extended representation, instances and algorithm components (low level heuristics) would be described together with the conventional model. In a sense, this is a natural evolution, given the decades of algorithm design guided by tests on benchmark instances. This combined representation would describe problem, instances and history in one comprehensive format.

All this may change our vision of how combinatorial problems are described, with constraint languages providing models for the problem in the conventional sense, together with information on how such problems could be solved. This information, in the conventional hyper-heuristics paradigm, is specified in terms of low level heuristics, but predominantly provided declaratively by the research community.

5.5 Future Directions: The Role of Ontologies

Ontologies codify knowledge in order to drive (traditionally formalized) reasoning processes. Their first recorded use dates back to Aristotle [40]. In computer science, the graph-based semantic nets of Quillan and Simmons [32] and Minsky's frame systems provided a foundational definition of entities in terms of specialization and part/whole relations [26]. These approaches subsequently spawned a multiplicity of variants (e.g. [20]). The previous section framed knowledge representation for hyper-heuristics in the vocabulary of constraint satisfaction. By expressing constraints as relations, such representations clearly have an equivalent representation as graphs, hypergraphs or RDF-triples. The use of an ontology for scheduling and routing problems can be seen in Martin et al. [25]. Recently, the Resource Description Framework (RDF) of the Semantic Web [3] has emerged as a common ontological basis for knowledge exchange. One important feature of web ontologies is that knowledge can be hierarchically constructed by referencing (the definitions for) other knowledge elements through a web-hosted URI.

What makes this relevant for hyper-heuristics is the associated support for the discovery, aggregation and substitution of uniquely-identifiable knowledge elements in the form of problem and algorithm descriptions. The goal then is that practitioner activity moves from 'under-the-hood' software development to the use of tools to hybridize pre-existing declarative specifications or else tweak constraints. Ontologies provide further support for the large-scale vision of hyper-heuristics, consisting of online data repositories containing discovered rules, patterns and constraints which describe good solution approaches. A suitable choice of knowledge representation elements (e.g. based on XCSP as above and/or other interoperability standards such as OpenMath [1]) can form the basis of community investment in such cross-domain learning tools. This is to be contrasted with the more isolated and domain-specific development that typically takes place in today's research settings.

5.6 Conclusion

We have traced the historical development of hyper-heuristics and highlighted the motivating division of responsibility: hyper-heuristic researchers are responsible for devising methods which work well across domains, with the goal of allowing practitioners to invest minimal effort in modelling a new domain. A requirement for ‘lifelong, cross-domain learning’ is strongly implied by this division of responsibility: when provided with the definition of a new domain, a hyper-heuristic must be able to produce effective solutions in this domain without significant practitioner expertise or intervention.

As part of a wider community initiative, we therefore argue for a polar stance to that of the prevailing view of hyper-heuristics: instead of imposing a ‘maximally restrictive’ interface between problem-domain and hyper-heuristic solver, we propose that it is vital to make problem domain (‘analytic’) and solution trajectory (‘empirical’) information available to the solver via some ‘universal’ knowledge exchange format. The wider possibilities then include:

- The extension of the algorithm selection problem to include ‘analytic’ information as part of the mapping process.
- The availability of arbitrarily rich features for machine learning approaches.
- The creation of a library of declarative descriptions of domains via a constraint language, more easily customized for a new domain than program code.

To coordinate these varied activities, a wider community initiative is in progress to promote an architectural vision of ‘Metaheuristics in the Large’ [47].

References

1. <http://www.openmath.org>. Online. Accessed Oct 2015
2. Abscon, an XCSP v2.0 solver. <http://www.cril.univ-artois.fr/~lecoutre/software.html>. Online. Accessed Oct 2015
3. G. Antoniou, F. Van Harmelen, *A Semantic Web Primer* (MIT, Cambridge, 2004)
4. S. Asta, E. Özcan, A tensor-based selection hyper-heuristic for cross-domain heuristic search. *Inf. Sci.* **299**, 412–432 (2015)
5. R. Battiti, Reactive search: toward self-tuning heuristics, in *Modern Heuristic Search Methods*, ed. by V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, G.D. Smith (Wiley, Chichester, 1996), pp. 61–83
6. U. Benlic, J.-K. Hao, A study of adaptive perturbation strategy for iterated local search, in *Evolutionary Computation in Combinatorial Optimization - 13th European Conference, EvoCOP 2013, Proceedings*, Vienna, 3–5 April 2013, pp. 61–72
7. A.E.I. Brownlee, J. Swan, E. Özcan, A.J. Parkes, Hyperion²: a toolkit for {Meta-, Hyper-} heuristic research, in *Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion*, GECCO Comp ’14 (ACM, New York, 2014), pp. 1133–1140
8. E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, S. Schulenburg, Hyper-heuristics: an emerging direction in modern search technology, in *Handbook of Metaheuristics* (Springer, Berlin, 2003), pp. 457–474

9. E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, J.R. Woodward, A classification of hyper-heuristic approaches, in *Handbook of Metaheuristics* (Springer, Berlin, 2010), pp. 449–468
10. E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: a survey of the state of the art. *J. Oper. Res. Soc.* **64**(12), 1695–1724 (2013)
11. K. Chakhlevitch, P.I. Cowling, Hyperheuristics: recent developments, in *Adaptive and Multilevel Metaheuristics*, ed. by C. Cotta, M. Sevaux, K. Sörensen. *Studies in Computational Intelligence* (Springer, Berlin, 2008), pp. 3–29
12. P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, in *Practice and Theory of Automated Timetabling III*. *Lecture Notes in Computer Science*, vol. 2079, ed. by E. Burke, W. Erben (Springer, Berlin, Heidelberg, 2001), pp. 176–190
13. L. De Raedt, T. Guns, S. Nijssen, Constraint programming for data mining and machine learning, in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)* (2010)
14. H.L. Fang, P. Ross, D. Corne, A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems, in *Proceedings of the Fifth International Conference on Genetic Algorithms* (Morgan Kaufmann, San Mateo, CA, 1993), pp. 375–382
15. H.L. Fang, P. Ross, D. Corne, A promising hybrid GA/heuristic approach for open-shop scheduling problems, in *Proceedings of the 11th Conference on Artificial Intelligence* (1994), pp. 590–594
16. H. Fisher, G.L. Thompson, Probabilistic learning combinations of local job-shop scheduling rules, in *Industrial Scheduling*, ed. by J.F. Muth, G.L. Thompson (Prentice-Hall, Upper Saddle River, NJ, 1963), pp. 225–251
17. M. Gagliolo, J. Schmidhuber, Learning dynamic algorithm portfolios. *Ann. Math. Artif. Intell.* **47**(3–4), 295–328 (2006)
18. F. Glover, M. Laguna, *Tabu Search* (Kluwer, Norwell, MA, 1997)
19. C.P. Gomes, B. Selman, Algorithm portfolios. *Artif. Intell.* **126**(1–2), 43–62 (2001)
20. T.R. Gruber, Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum. Comput. Stud.* **43**(5), 907–928 (1995)
21. B.A. Huberman, R.M. Lukose, T. Hogg, An economics approach to hard computational problems. *Science* **275**(5296), 51–54 (1997)
22. M. Hyde, E. Özcan, E.K. Burke, Multilevel search for evolving the acceptance criteria of a hyper-heuristic, in *Proceedings of the 4th Multidisciplinary International Conference on Scheduling: Theory and Applications* (2009), pp. 798–801
23. G. Kendall, E. Soubeiga, P. Cowling, Choice function and random hyperheuristics, in *Proceedings of the fourth Asia-Pacific Conference on Simulated Evolution And Learning, SEAL* (Springer, Berlin, 2002), pp. 667–671
24. D.B. Lenat, EURISKO: a program that learns new heuristics and domain concepts. *Artif. Intell.* **21**(1–2), 61–98 (1983)
25. S. Martin, D. Ouelhadj, P. Smet, G.V. Berghe, E. Özcan, Cooperative search for fair nurse rosters. *Expert Syst. Appl.* **40**(16), 6674–6683 (2013)
26. M. Minsky, A framework for representing knowledge. Technical Report, Cambridge, MA (1974)
27. J. Mostow, A.E. Prieditis, Discovering admissible heuristics by abstracting and optimizing: a transformational approach, in *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI'89*, San Francisco, CA (Morgan Kaufmann, San Mateo, CA, 1989), pp. 701–707
28. G. Ochoa, M. Hyde, T. Curtois, A. Vazquez-Rodriguez, J. Walker, M. Gendreau, B. Kendall, G. McCollum, A.J. Parkes, S. Petrovic et al., Hyflex: a benchmark framework for cross-domain heuristic search, in *Evolutionary Computation in Combinatorial Optimization* (Springer, Berlin, 2012), pp. 136–147
29. E. Özcan, B. Bilgin, E.E. Korkmaz, Hill climbers and mutational heuristics in hyperheuristics, in *Parallel Problem Solving from Nature - PPSN IX*. *Lecture Notes in Computer Science*, vol. 4193 (Springer, Berlin, Heidelberg, 2006), pp. 202–211

30. A.J. Parkes, E. Özcan, D. Karapetyan, A software interface for supporting the application of data science to optimisation, in *Learning and Intelligent Optimization*. Lecture Notes in Computer Science, vol. 8994 (Springer, Berlin, 2015), pp. 306–311
31. R. Qu, E.K. Burke, B. McCollum, Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. *Eur. J. Oper. Res.* **198**(2), 392–404 (2009)
32. R. Quillan, *A Notation for Representing Conceptual Information: An Application to Semantics and Mechanical English Paraphrasing* (Systems Development Corporation, Santa Monica, CA, 1963)
33. J.R. Rice, The algorithm selection problem, in *Advances in Computers*, vol. 15 (Elsevier, New York, 1976), pp. 65–118
34. P. Ross, Hyper-heuristics, in *Search Methodologies*, ed. by E.K. Burke, G. Kendall (Springer US, New York, 2005), pp. 529–556
35. P. Ross, S. Schulenburg, J.G. Marín-Blázquez, E. Hart, Hyper-heuristics: learning to combine simple heuristics in bin-packing problems, in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, San Francisco (Morgan Kaufmann, San Mateo, CA, 2002), pp. 942–948
36. P. Ross, J.G. Marín-Blázquez, S. Schulenburg, E. Hart, Learning a procedure that can solve hard bin-packing problems: a new GA-based approach to hyper-heuristics, in *Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation: Part II*, GECCO'03 (Springer, Berlin, Heidelberg, 2003), pp. 1295–1306
37. O. Roussel, C. Lecoutre, XML representation of constraint networks: format XCSP 2.1. CoRR (2009). abs/0902.2362
38. K. Sörensen, F.W. Glover, Metaheuristics, in *Encyclopedia of Operations Research and Management Science* (Springer US, New York, 2013), pp. 960–970
39. E. Soubeiga, *Development and Application of Hyperheuristics to Personnel Scheduling*. Ph.D. thesis, School of Computer Science, University of Nottingham (2003)
40. J.F. Sowa, *Knowledge Representation: Logical, Philosophical and Computational Foundations* (Brooks/Cole Publishing, Pacific Grove, CA, 2000)
41. R.H. Storer, S.D. Wu, R. Vaccari, New search spaces for sequencing problems with application to job shop scheduling. *Manag. Sci.* **38**(10), 1495–1509 (1992)
42. R.H. Storer, S.D. Wu, R. Vaccari, Problem and heuristic space search strategies for job shop scheduling. *ORSA J. Comput.* **7**(4), 453–467 (1995)
43. J. Swan, N. Burles, Templar - a framework for template-method hyper-heuristics, in *Genetic Programming*. Lecture Notes in Computer Science, vol. 9025, ed. by P. Machado et al. (Springer, Berlin, 2015), pp. 205–216
44. J. Swan, E. Özcan, G. Kendall, Hyperion - a recursive hyper-heuristic framework, in *Learning and Intelligent Optimization*, ed. by C. Coello. Lecture Notes in Computer Science, vol. 6683 (Springer, Berlin, Heidelberg, 2011), pp. 616–630
45. J. Swan, M. Edjvet, E. Özcan, Augmenting metaheuristics with rewriting systems. Technical Report CSM-197, Computing Science and Mathematics, University of Stirling, Stirling FK9 4LA (2014)
46. J. Swan, J.R. Woodward, E. Özcan, G. Kendall, E.K. Burke, Searching the hyper-heuristic design space. *Cogn. Comput.* **6**(1), 66–73 (2014)
47. J. Swan, S. Adriaensen, M. Bishr, E.K. Burke, J.A. Clark, P. De Causmaecker, J. Durillo, K. Hammond, E. Hart, C.G. Johnson, Z.A. Kocsis, B. Kovitz, K. Krawiec, S. Martin, J.J. Merelo, L.L. Minku, E. Özcan, G.L. Pappa, E. Pesch, P. Garcia-Sánchez, A. Schaerf, K. Sim, J. Smith, T. Stützle, S. Voß, S. Wagner, X. Yao, A research agenda for metaheuristic standardization, in *MIC 2015: The XI Metaheuristics International Conference* (2015)
48. J. Woodward, A. Parkes, G. Ochoa, A mathematical formalization of hyper-heuristics. Workshop on Hyper-Heuristics - Automating the Heuristic Design Process (2008) <http://www.cs.stir.ac.uk/~jrw/publications/defHH.pdf>. Online. Accessed 21 Oct 2015
49. J. Woodward, J. Swan, S. Martin, The ‘Composite’ design pattern in metaheuristics, in *Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion*, GECCO Comp '14, New York, 2014, pp. 1439–1444
50. XCSP v3.0. <http://www.xcsp.org/series.html>. Online. Accessed Oct 2015

Chapter 6

POSL: A Parallel-Oriented Metaheuristic-Based Solver Language

Alejandro REYES-Amaro, Eric Monfroy, and Florian Richoux

Abstract For a couple of years, all processors in modern machines are multi-core. Massively parallel architectures, so far reserved for super-computers, become now available to a broad public through hardware like the Xeon Phi or GPU cards. This architecture strategy has been commonly adopted by processor manufacturers, allowing them to stick with Moore’s law. However, this new architecture implies new ways to design and implement algorithms to exploit its full potential. This is in particular true for constraint-based solvers dealing with combinatorial optimization problems. Here we propose a Parallel-Oriented Solver Language (POSL, pronounced “puzzle”), a new framework to build interconnected meta-heuristic based solvers working in parallel. The novelty of this approach lies in looking at solver as a set of components with specific goals, written in a parallel-oriented language based on operators. A major feature in POSL is the possibility to share not only information, but also behaviors, allowing solver modifications during runtime. Our framework has been designed to easily build constraint-based solvers and reduce the developing effort in the context of parallel architecture. POSL’s main advantage is to allow solver designers to quickly test different heuristics and parallel communication strategies to solve combinatorial optimization problems, usually time-consuming and very complex technically, requiring a lot of engineering.

Keywords CSP • Meta-heuristic • Parallel • Inter-process communication • Language

6.1 Introduction

Combinatorial Optimization has strong applications in several fields, including machine learning, artificial intelligence, and software engineering. In some cases, the main goal is only to find a solution, like for *Constraint Satisfaction Problems (CSP)*.

A. REYES-Amaro (✉) • E. Monfroy • F. Richoux
LINA Inria–TASC, Université de Nantes, 2 rue de la Houssinière, Nantes, France
e-mail: alejandro.reyes@univ-nantes.fr;
eric.monfroy@univ-nantes.fr; florian.richoux@univ-nantes.fr

A solution will be an assignment of variables satisfying the constraints set. In other words: finding one feasible solution.

CSPs find a lot of applications in the industry, implying the development of many methods to solve them. Meta-heuristics techniques have shown themselves to be effective for solving CSPs, but in most industrial cases the search space is huge enough to be intractable. However, recent advances in computer architecture are leading us toward massively *multi/many-core* computers, opening a new way to find solutions for these problems in a more feasible manner, reducing search time. Adaptive Search [5] is an efficient methods showing very good performances scaling to hundreds or even thousands of cores, using a multi-walk local search method. For this algorithm, an implementation of a cooperative multi-walks strategy has been published in [9]. These works have shown the efficiency of multi-walk strategy, that is why we have oriented POSL towards this parallel scheme.

In the last years, a lot of efforts have been made in parallel constraint programming. In this field, the inter-process communication for solver cooperation is one of the most critical issues. Pajot and Monfroy [11] presents a paradigm that enables the user to properly separate strategies combining solver applications in order to find the desired result, from the way the search space is explored. *Meta-S* is an implementation of a theoretical framework proposed in [6], which allows to tackle problems, through the cooperation of arbitrary domain-specific constraint solvers. POSL provides a mechanism of creating solver-independent communication strategies, making easy the study of solving processes and results. Creating solvers implementing different solution strategies can be complex and tedious. In that sense POSL gives the possibility of prototyping communicating solvers with few efforts.

In Constraint Programming, many researches focus on fitting and improving existing algorithms for specific problems. However, it requires a deep study to find the right algorithm for the right problem. HYPERION [3] is a Java framework for meta- and hyper-heuristics built with the principle of interoperability, generality by providing generic templates for a variety of local search and evolutionary computation algorithms and efficiency, allowing rapid prototyping with the possibility of reusing source code. POSL aims to offer the same advantages, but provides also a mechanism to define communication protocols between solvers.

In this chapter we present POSL, a framework for easily building many and different cooperating solvers based on coupling four fundamental and independent components: *operation modules*, *open channels*, the *computation strategy* and *communication channels* or *subscriptions*. Recently, the hybridization approach leads to very good results in constraint satisfaction [14]. *ParadisEO* is a framework to design parallel and distributed hybrid meta-heuristics showing very good results [4]. It includes a broad range of reusable features to easily design evolutionary algorithms and local search methods. Our framework POSL focuses only in local search methods, but is designed to execute in parallel sets of different solvers, with and/or without communication, since the solver's components can be combined by using operators.

POSL provides, through a simple operator-based language, a way to create a *computation strategy*, combining already defined *components* (*operation modules* and *open channels*). A similar idea was proposed in [7] without communication, introducing an evolutionary approach that uses a simple composition operator to automatically discover new local search heuristics for SAT and to visualize them as combinations of a set of building blocks. Another interesting idea is proposed in TEMPLAR, a framework to generate algorithms changing predefined components using hyper-heuristics methods [13]. In the last phase of the coding process with POSL, solvers can be connected each others, depending on the structure of their *open channels*, and this way, they can share not only information, but also their behavior, by sharing their *operation modules*. This approach makes the solvers able to evolve during the execution.

Before ending this chapter with a brief conclusion and future works, we present some results obtained by using POSL to solve some instances of the *Social Golfers Problem*.

6.2 POSL Parallel Solvers

POSL proposes a solver construction platform following different stages. First of all, the solver algorithm is modeled by decomposing it into small pieces/modules of computation. After that, they are implemented as separated *functions*. We name them *operation module*. The next step is to decide what information is interesting to receive from other solvers. This information is encapsulated into other objects called *open channels*, allowing data transmission among solvers. In a third stage, a generic strategy is coded through POSL, using the mentioned components in the previous stages, allowing not only the information exchange, but also to execute the components in parallel. This will be the solver's backbone. Finally, solvers are defined by instantiating and connecting the *strategy*, *operation modules* and *open channels*, and by connecting them each others. The next subsections explain in details each of these steps.

6.2.1 Operation Module

An *operation module* is the most basic and abstract way to define a piece of computation. It can be dynamically replaced by or combined with other *operation modules*, since they can be sheared among solvers working in parallel. This way, the solver can mutate its behavior during execution.

An *operation module* receives an input, executes an internal algorithm and gives an output. They are joined through *computation strategies*.

Definition 6.1. Operation Module An *operation module* Om is a mapping defined by:

$$Om : D \rightarrow I \tag{6.1}$$

D and I can be either a set of configurations, or set of sets of configurations, or a set of values of some data type, etc.

Consider a local search meta-heuristic solver. One of its *operation modules* can be the function returning the set of configurations composing the neighborhood of a given configuration:

$$Om_{neighborhood} : D_1 \times D_2 \times \dots \times D_n \rightarrow 2^{D_1 \times D_2 \times \dots \times D_n}$$

where D_i represents the definition domains of each variable of the input configuration.

6.2.2 Open Channels

Open Channels are the solver's components in charge of the information reception in the communication between solvers. They can interact with *operation modules*, depending on the *computation strategy*. *Open Channels* play the role of outlets, allowing solvers to be connected and to share information.

An *open channel* can receive two types of information, always coming from an external solver: data or *operation modules*. It is important to notice that when we are talking about sending/receiving *operation modules*, we mean sending/receiving only required information to identify it and being able to instantiate it.

In order to distinguish between the two different types of *open channels*, we will call Data Open Channel the *open channel* responsible for the data reception, and Object Open Channel the one responsible for the reception and instantiation of *operation modules*.

Definition 6.2. Data Open Channel A *Data Open Channel* Ch is a component that produces a mapping defined as follows:

$$Ch : U \rightarrow I \tag{6.2}$$

It returns the information I coming from an external solver, no matter what the input U is.

Definition 6.3. Object Open Channel If we denote by \mathbb{M} the space of all the *operation modules* defined by Definition 6.1, then an *Object Open Channel* Ch is a component that produces an *operation module* coming from an external solver as follows:

$$Ch : \mathbb{M} \rightarrow \mathbb{M} \tag{6.3}$$

Due to the fact that *open channels* receive information coming from outside and have no control on them, it is necessary to define the *NULL* information, to denote

the absence of any information. If a Data Open Channel receives a piece of information, it is returned automatically. If a Object Open Channel receives an *operation module*, the latter is instantiated and executed with the *open channel*'s input, and its result is returned. In both cases, if no available information exists (no communications are performed), the *open channel* returns the *NULL* object.

6.2.3 Computation Strategy

The *computation strategy* is the solver's backbone: it joins *operation modules* and *open channels* in a coherent way, while remaining independent from them. Through the *computation strategy* we can decide also what information to sent to other solvers.

The *computation strategy* is an operator-based language, that we define as a free-context grammar as follows:

Definition 6.4. POSL's Grammar $G_{POSL} = (\mathbf{V}, \Sigma, \mathbf{S}, \mathbf{R})$, where:

1. $\mathbf{V} = \{CM, OP\}$ is the set of *variables*,
2. $\Sigma = \left\{ om, och, be, [,], \llbracket, \rrbracket_p, (,), \{, \}, \langle, \rangle^m, \rangle^o \mapsto, \odot, \rho, \vee, \mathbf{M}, \mathbf{m}, \downarrow \right\}$
is the set of *terminals*,
3. $\mathbf{S} = \{CM\}$ is the set of *start variables*,
4. and the set of *rules* $\mathbf{R} =$

$$CM \rightarrow om|och|(\langle om \rangle^o | \langle om \rangle^m | [OP] | \llbracket OP \rrbracket_p$$

$$OP \rightarrow CM \mapsto CM$$

$$OP \rightarrow CM \mapsto (be) \{CM; CM\}$$

$$OP \rightarrow CM \odot (be) \{CM\}$$

$$OP \rightarrow CM \rho CM | CM \vee CM | CM \mathbf{M} CM | CM \mathbf{m} CM | CM \downarrow CM$$

We would like to explain some of the concepts presented in Definition 6.4:

- The variable *CM*, as well as *OP* are two entities very important in the language, as can be seen in the grammar. We name them *compound module* and *operator* respectively.
- The terminals *om* and *och* represent an *operation module* and an *open channel* respectively,
- The terminal *be* is a boolean expression.
- The terminals $[,]$, \llbracket, \rrbracket_p are symbols for grouping and defining the way of how the involved *compound modules* are executed. Depending on the nature of the operator, they can be executed sequentially or in parallel:

1. [OP]: The involved operator is executed sequentially.
 2. $\llbracket \text{OP} \rrbracket_p$: The involved operator is executed in parallel if and only if OP supports parallelism. Otherwise, an exception is threw.
- The terminals (and) are symbols for grouping the boolean expression in some operators.
 - The terminals { and } are symbols for grouping *compound modules* in some operators.
 - The terminals $(\cdot)_m, (\cdot)_o$, are operators to send information to other solvers (explained bellow).
 - The rest of terminals are POSL operators.

6.2.3.1 POSL Operators

In this section we briefly present operators provided by POSL to code the *computation strategy*. A formal presentation of POSL's specification is available in [12].

Op. 1 : **Operator Sequential Execution:** the operation $M_1 \mapsto M_2$ represents a *compound module* as result of the execution of M_1 followed by M_2 . This operator is an example of an operator that does not support the execution of its involved *compound modules* in parallel, because the input of the second *compound module* is the output of the first one.

Op. 2 : **Operator Conditional Sequential Execution:** the operation $M_1 \mapsto (\langle \text{cond} \rangle) \{M_2, M_3\}$ represents a *compound module* as result of the sequential execution of M_1 followed by M_2 if $\langle \text{cond} \rangle$ is **true** or by M_3 otherwise.

Op. 3 : **Operator Cyclic Execution:** the operation $\cup (\langle \text{cond} \rangle) \{I_1\}$ represents a *compound module* as result of the sequential execution of I_1 repeated while $\langle \text{cond} \rangle$ remains **true**.

Op. 4 : **Operator Random Choice:** the operation $M_1 \circlearrowleft \rho M_2$ represents a *compound module* that executes and returns the output of M_1 depending on the probability ρ , or M_2 following $(1 - \rho)$

Op. 5 : **Operator Not NULL Execution:** the operation $M_1 \circlearrowright M_2$ represents a *compound module* that executes M_1 if it is not *NULL* or M_2 otherwise.

Op. 6 : **Operator MAX:** the operation $M_1 \circlearrowright M_2$ represents a *compound module* that returns the maximum between the outputs of modules M_1 and M_2 (tacking into account some order criteria).

Op. 7 : **Operator MIN:** the operation $M_1 \circlearrowleft M_2$ represents a *compound module* that returns the minimum between the outputs of modules M_1 and M_2 (tacking into account some order criteria).

Op. 8 : **Operator Speed:** the operation $M_1 \circlearrowdown M_2$ represents a *compound module* that returns the output of the *module* ending first.

In Fig. 6.1 we present a simple example of how to combine *modules* using POSL operators introduced above. Algorithm 1 shows the corresponding code. In this example we show four *operation modules* being part of a *compound module* representing a dummy local search method. In this example:

- M_1 : generates a random configuration.
- M_2 : computes a neighborhood of a given configuration by selecting a random variable and changing its value.
- M_3 : computes a neighborhood of a given configuration by selecting K random variables and changing their values.
- M_4 : selects, from a set of configurations, the one with the smallest cost, and stores it.

Here, the *operation module* M_2 is executed with probability ρ , and M_3 is executed with probability $(1 - \rho)$. This operation is repeated a number N of times ($< \text{stop_cond} >$).

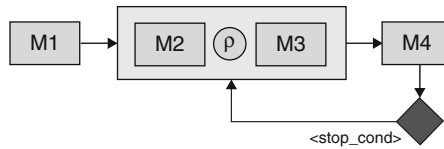


Fig. 6.1 Un example of a basic solver using POSL

Algorithm 1 POSL code for Fig. 6.1

$$M_1 \mapsto [\cup (\text{loops} < N) \{ [M_2 \text{ } \rho \text{ } M_3] \mapsto M_4 \}]$$

In Algorithm 1, *loops* represent the number of iterations performed by the operator.

+2

Op. - 8 : **Operator Sending:** allows us to send two types of information to other solvers:

1. The operation $\langle M \rangle^o$ represents a *compound module* that executes the *compound module* M and sends its output
2. The operation $\langle M \rangle^m$ represents a *compound module* that executes the *compound module* M and sends M itself

Algorithm 2 POSL code for Fig. 6.2 case (1)

$$M_1 \mapsto \langle M \rangle^o \mapsto M_2$$

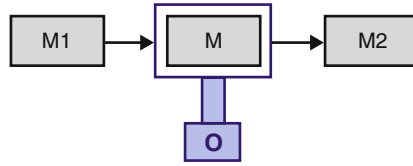


Fig. 6.2 Sending Information Operator

Algorithm 3 POSL code for Fig. 6.2 case (2)

$$M_1 \mapsto \langle M \rangle^m \mapsto M_2$$

Algorithms 2 and 3 show POSL’s code corresponding to Fig. 6.2 for both cases: (a) sending the result of the execution of the *operation module* M , or (b) sending the *operation module* M itself.

This operation is very useful in terms of sharing behaviors between solvers. Figure 6.3 shows another example, where we can combine an *open channel* with the *operation module* M_2 through the operator \odot . In this case, the *operation module* M_2 will be executed as long as the *open channel* remains *NULL*, i.e. there is no *operation module* coming from outside. This behavior is represented in Fig. 6.3 by red lines. If some *operation module* has been received by the *open channel*, it is executed instead of the *operation module* M_2 , represented in Fig. 6.3 by blue lines.

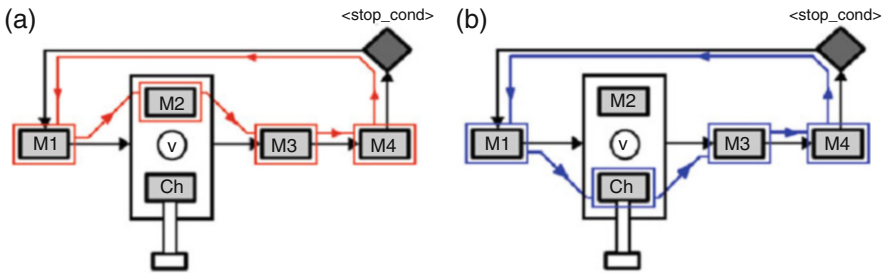


Fig. 6.3 Two different behaviors in the same solver. (a) The solver executes his own *operation module* if no information is received through the *open channel*. (b) The solver executes the *operation module* coming from an external solver

In this stage, and using these operators, we can create the algorithm managing different components to find the solution of a given problem. These algorithms are fixed, but generic w.r.t. their components (*operation modules* and *open channels*). It means that we can build different solvers using the same strategy, but instantiating it with different components, as long as they have the right input/output signature.

To define a *computation strategy* we use the environment presented in Algorithm 6, where M_i and Ch_i represent the types of the *operation modules* and the types of the *open channels* used by the *computation strategy* St . Between brackets, the field `< ...computation strategy... >` corresponds to POSL code based on operators combining already declared *modules*.

Algorithm 4 *Computation strategy definition*

```

St ← strategy
oModule  $M_1, M_2, \dots, M_n$ 
oChannel  $Ch_1, Ch_2, \dots, Ch_m$ 
{
< ...computation strategy... >
}

```

Algorithm 5 *Solver definition*

```

solverk ← solver
{
cStrategy St
oModule  $m_1, m_2, \dots, m_n$ 
oChannel  $ch_1, ch_2, \dots, ch_m$ 
}

```

6.2.4 Solver Definition

With *operation modules*, *open channels* and *computation strategy* defined, we can create solvers by instantiating the declared components. POSL provides an environment to this end, presented in Algorithm 6, where m_i and ch_i represent the instances of the *operation modules* and the instances of the *open channels* to be passed by parameters to the *computation strategy* St .

6.2.5 Communication Definition

Once we have defined our solver strategy, the next step is to declare communication channels, i.e. connecting the solvers each others. Up to here, solvers are disconnected, but they have everything to establish the communication. In this last stage, POSL provides to the user a platform to easily define cooperative *meta-strategies* that solvers must follow.

The communication is established by following the next rules guideline:

1. Each time a solver sends any kind of information by using the operator $(\cdot)^o$ or $(\cdot)^m$, it creates a *communication jack*
2. Each time a solver uses an *open channel* into its definition, it creates a *communication outlet*
3. Solvers can be connected each others by creating *subscriptions*, connecting *communication jacks* with *communication outlet* (see Fig. 6.4).

With the operator (\cdot) we have access to *operation modules* sending information and to the *open channel's* names in a solver. For example: $Solver_1 \cdot M_1$ provides access to the *operation module* M_1 in $Solver_1$ if and only if it is affected by the operator $(\cdot)^o$ (or $(\cdot)^m$), and $Solver_2 \cdot Ch_2$ provides access to the *open channel* Ch_2 in $Solver_2$. Tacking this into account, we can define the *subscriptions*.

Definition 6.5. Let two different solvers $Solver_1$ and $Solver_2$ be. Then, we can connect them through the following operation:

$$Solver_1 \cdot M_1 \rightsquigarrow Solver_2 \cdot Ch_2$$

The connection can be defined if and only if:

1. $Solver_1$ has an *operation module* called M_1 encapsulated into an operator $(\cdot)^o$ or $(\cdot)^m$.
2. $Solver_2$ has an *open channel* called Ch_2 receiving the same type of information sent by M_1 .

Definition 6.5 only gives the possibility to define static communication strategies. However, our goal is to develop this subject until obtaining operators more expressive in terms of communication between solvers, to allow dynamic modifications of communication strategies, that is, having such strategies adapting themselves during runtime.

6.3 A POSL Solver

In this section we explain the structure of a POSL solver created by using the operators-based language provided, to solve some instances of the *Social Golfers Problem* (SGP). It consists to schedule $n = g \times p$ golfers into g groups of p players every week for w weeks, such that two players play in the same group at most once. An instance of this problem can be represented by the triple $g - p - w$.

We choose one of the more classic solution methods for combinatorial problems: local search meta-heuristics algorithms. These algorithms have a common structure: they start by initializing some data structures (e.g. a *tabu list* for *Tabu Search* [8], a *temperature* for *Simulated Annealing* [10], etc.). Then, an initial configuration s is generated (either randomly or by using heuristic). After that, a new configuration s^* is selected from the neighborhood $V(s)$. If s^* is a solution for the problem P , then

the process stops, and s^* is returned. If not, the data structures are updated, and s^* is accepted or not for the next iteration, depending on some criterion (e.g. penalizing features of local optimums, like in *Guided Local Search* [2]).

Restarts are classic mechanisms to avoid becoming trapped in local minimum. They are triggered if no improvements are done or by a timeout.

Operation Modules composing each solver of the POSL-solver are described below:

1. Generate a configuration s .
2. Define the neighborhood $V(s)$
3. Select $s^* \in V(s)$. In every case for this experiment the selection criteria is to choose the first configuration improving the cost.
4. Evaluate an acceptance criteria for s^* . In every case for this experiment the acceptance criteria is to choose always the configuration with less cost.

For this particular experiment we have created three different solvers (see Fig. 6.4):

1. **Solver 1:** A solver sending the best configuration every K iterations (sender solver). It sends the found configuration to the solver it is connected with. Algorithm 5 shows its computation strategy.
2. **Solver 2:** A solver receiving the configuration coming from a sender solver (**Solver 1**). It takes the received configuration, if its current configuration's cost is not better than the received configuration's cost, and takes a decision. This solver receives the configuration through an *open channel* joined to the *operation module* M_3 with the operator \odot . Algorithm 7 shows its computation strategy.
3. **Solver 3:** A simple solver without communication at all. This solver does not communicate with any other solver, i.e. it searches the solution into an independent walk though the search space. Algorithm 5 shows its computation strategy.

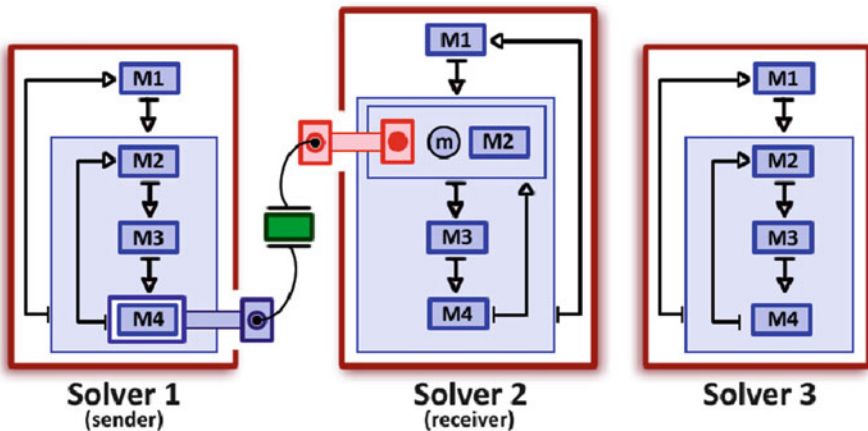


Fig. 6.4 Three solvers composing the POSL-solver

6.3.1 Connecting Solvers

After the instantiation of each *operation module*, the next step is to connect the solvers (*sender* with *receiver*), by using the proper operator. If one solver Σ_1 (*sender*) sends some information and some other solver Σ_2 (*receiver*) is able to receive it through an *open channel*, then they can be connected as the Algorithm 1 shows.

Algorithm 6 POSL code for solver 1 in Fig. 6.4

```

S1 ← strategy      /* ITR → number of iterations */
oModule : M1, M2, M3, M4
{
  [⊙ (ITR%30){M1 ↦ [⊙ (ITR%300){M2 ↦ M3 ↦ {M4}]}]}
}

```

Algorithm 7 POSL code for solver 2 in Fig. 6.4

```

S2 ← strategy      /* ITR → number of iterations */
oModule : M1, M2, M3, M4
oChannel : Ch1
{
  [⊙ (ITR%30){M1 ↦ [⊙ (ITR%300)
  {M2 ↦ [Ch1 ⊙ M3] ↦ M4}]}]}
}

```

Algorithm 8 POSL code for solver 3 in Fig. 6.4

```

S3 ← strategy      /* ITR → number of iterations */
oModule : M1, M2, M3, M4
{
  [⊙ (ITR%30){M1 ↦ [⊙ (ITR%300){M2 ↦ M3 ↦ M4}]}]}
}

```

Algorithm 9 Inter-solvers communication definition

```

Σ1 · M4 ∼ ∼ Σ2 · Ch1

```

6.4 Results

We ran experiments to study the behavior of POSL's solvers in different scenarios solving instances of the Social Golfers Problem. For that reason we classified runs taking into account the composition of POSL¹ solvers:

- Without communication: we use a set of **solvers 3** without communication.
- Some communicating solvers: some of the solvers are **solvers 3** without communication, the others are couples of connected solvers (**solver 1** and **solver 2**)
- All communicating solvers: we use a set of couples of connected solvers (**solver 1** and **solver 2**).

Our first experiment uses our desktop computer (Intel ®Core™i7 (2.20 GHz) with 16 Gb RAM), for solving instances of SGP with 1 (sequential), 4 and 8 cores. Results can be found in Table 6.1. In this table, as well as in Table 6.2, **C** are the numbers of used cores, **T** indicates the runtime in milliseconds, and **It.** the number of iterations. Values are the mean of 25 runs for each setup.

The other set of runs were performed on the server of our laboratory (Intel ®Xeon™E5-2680 v2 (10×4 cores, 2.80GHz)). Table 6.2 shows obtained results.

Table 6.1 Intel Core i7

Inst	C	No comm.		50% comm.		All comm.	
		T	It.	T	It.	T	It.
6-6-3	1	6089	159	–	–	–	–
	4	1500	109	1354	97	3512	181
	8	1980	83	2049	78	5323	113
7-7-3	1	17,243	831	–	–	–	–
	4	6082	208	5850	170	13,094	270
	8	6125	136	5975	124	13,864	219
8-8-3	1	32,042	428	–	–	–	–
	4	23,358	270	22,512	222	56,740	340
	8	19,309	126	19,925	121	28,036	144
9-9-3	1	198,450	1516	–	–	–	–
	4	94,867	662	91,556	517	102,974	596
	8	102,629	394	98,060	335	126,799	466

Bold values: Best (lowest) result considering the number of processors used in the experiment

Results show how the parallel multi-walk strategy increases the probability of finding the solution within a reasonable time, when compared to the sequential scheme. Thanks to POSL it was possible to test different solution strategies easily and quickly. With the *Intel Xeon* server we were able to test seven strategies, and with the desktop machine only 3, due to the limitation in the number of cores. Results suggest that strategies where there exist a lot of communication between solvers (sending or receiving information) are not good (sometimes is even worse than sequential). That is not only because their runtimes are higher, but also due to

¹ POSL source code is available in <https://github.com/alejandro-reyesamaro/POSL>.

Table 6.2 Intel Xeon

Inst	C	No comm.		15% comm.		25% comm.		30% comm.		50% comm.		75% comm.		All comm.	
		T	It.	T	It.	T	It.	T	It.	T	It.	T	It.	T	It.
6-6-3	1	2684	229	-	-	-	-	-	-	-	-	-	-	-	-
	10	1810	131	1636	107	1479	99	1634	107	1406	79	1532	91	3410	182
	20	1199	82	1094	75	964	70	1096	76	1124	78	1299	87	1769	101
	30	1214	75	1092	64	1010	68	1101	68	766	52	1366	85	1984	73
	40	1043	50	1063	49	1104	54	1299	58	1186	49	1462	63	1824	69
7-7-3	1	11,070	533	-	-	-	-	-	-	-	-	-	-	-	-
	10	6636	245	5992	189	5139	179	5456	177	6055	205	6398	197	8450	221
	20	2734	104	2880	102	2517	90	3028	111	2970	111	3465	124	4153	143
	30	3141	100	2864	91	1972	69	2312	79	2907	97	3028	82	3236	89
	40	2615	68	2810	70	2111	55	2984	71	2981	74	3636	79	3934	86
8-8-3	1	24,829	315	-	-	-	-	-	-	-	-	-	-	-	-
	10	17,652	193	17,067	168	16,008	163	16,167	161	16,624	147	21,244	185	27,248	226
	20	8430	102	8218	92	6197	77	7950	93	7962	92	8550	91	12,958	125
	30	7424	81	6439	66	6268	71	7413	80	7407	75	9806	89	10,420	90
	40	9700	75	10,068	76	9377	72	8983	68	9360	72	11,805	84	12,859	91
9-9-3	1	190,965	1315	-	-	-	-	-	-	-	-	-	-	-	-
	10	47,300	331	45,946	293	43,682	276	45,433	286	47,820	327	67,113	439	79,938	506
	20	28,193	200	25,370	178	24,936	161	24,786	169	28,369	194	30,147	203	33,610	232
	30	22,035	123	21,792	127	19,518	125	23,426	133	25,989	163	31,904	172	32,982	203
	40	27,669	125	26,030	116	24,196	112	28,284	125	26,405	118	32,464	149	34,316	140

Bold values: Best (lowest) result considering the number of processors used in the experiment

the fact that only a low percentage of the receivers solvers were able to find the solution before the others did. This result is not surprising, because inter-process communications imply overheads in the computation process, even with asynchronous communications. This phenomenon can be seen in Fig. 6.5, where it is analyzed the percentage mentioned above versus the numbers of running solvers.

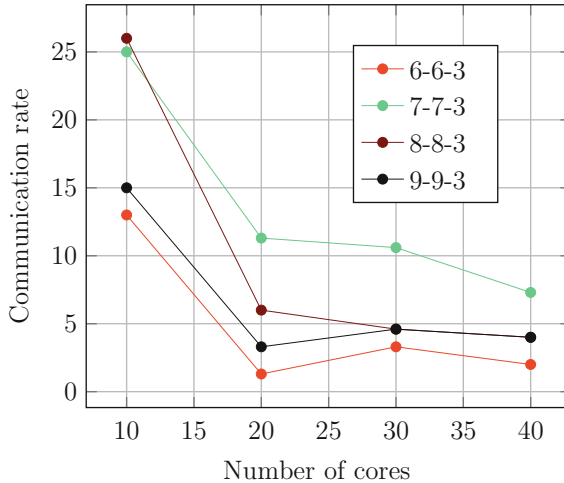


Fig. 6.5 Communication rate: % of solutions found by communicating solvers

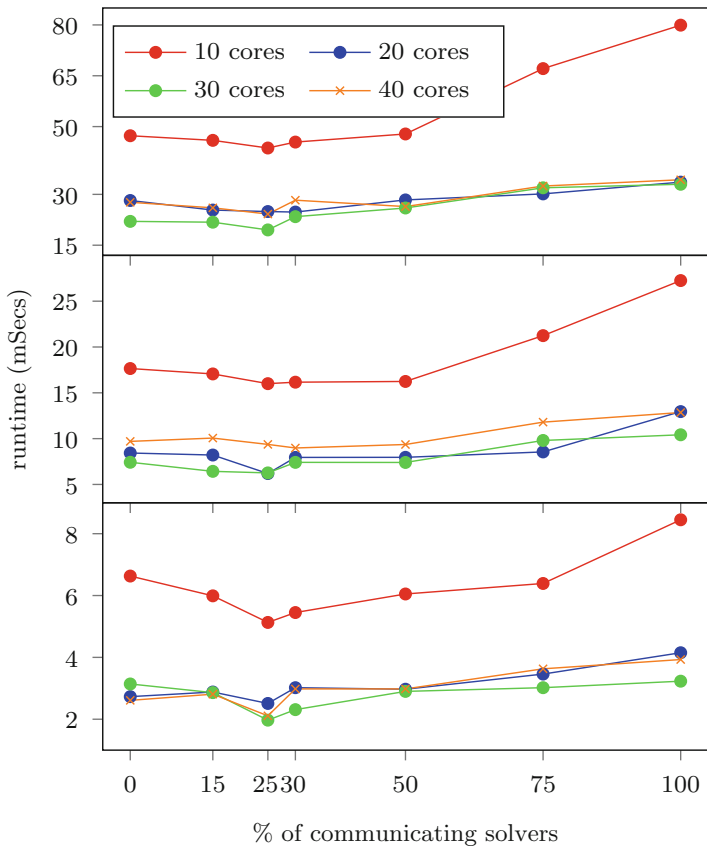


Fig. 6.6 Runtime means of instances 7-7-3, 8-8-3 and 9-9-3

When we face the problem of building a parallel strategy, it is necessary to find an equilibrium between the numbers of communicating solvers and the number of independent solvers. Indeed the communication cost is not negligible: it implies data reception, information interpretation, making decisions, etc.

Slightly better results were obtained with the strategy *25% Comm* when compared to those obtained with the rest, suggesting that the solvers cooperation can be a good strategy. In general, the results obtained using any of the afore mentioned strategies were significantly better than when using the *All Comm* strategy. Figure 6.6 shows for each instance, the runtime means using different numbers of cores.

The fact we send the best configuration found to other solvers has an impact on communication evaluations. If the percentage of communicating solvers is high and the communication manage to be effective, i.e. the receiver solver accepts the configuration for the next iteration, then we are losing a bit the *independent multi-walk*

effect in our solver, that is, most of the solvers are looking for a solution in the same search space area. However, this is not a problem: if a solver is trapped, a restart is performed. Determining what information to share and to not share among solvers has been few investigated and deserves a deep study.

In many cases, using all cores available did not improve the results. This phenomenon can be observed clearly in runs with communication, and one explanation can be the resulting overhead, which is way bigger. Another reason why we obtain these results can be the characteristic of the architecture, that is, in many cases, not uniform in terms of reachability between cores [1]. We can observe that, even if runtimes are not following a strict decreasing pattern when the number of cores increases, iterations do, suggesting once again that the parallel approach is effective.

With communications, the larger the problem, the more likely effective cooperations between processors are, although sometimes a decreasing pattern occurs while approaching the maximum number of cores, due to communication overheads and architecture limitations.

Before we perform these experiments, we compared runtimes between two solvers: one using an *operation module* to select a configuration from a computed neighborhood that selects the *first* configuration improving the current configuration's cost, and other selecting the *best* configuration among all configurations in the neighborhood. Smallest runtimes were obtained by the one selecting the *first* best configuration, and that is way we used this *operation module* in our experiments. It explains the fact that some solvers need more time to perform less iterations.

6.5 Conclusions

In this chapter we have presented POSL, a framework for building cooperating solvers. It provides an effective way to build solvers which exchange any kind of information, including other solver's behavior, sharing their *operation modules*. Using POSL, many different solvers can be created and ran in parallel, using only one generic strategy, but instantiating different *operation modules* and *open channels* for each of them.

It is possible to implement different communication strategies, since POSL provides a layer to define *communication channels* connecting solvers dynamically using *subscriptions*.

At this point, the implementation of POSL remains in progress, in which our principal task is creating a design as general as possible, allowing to add new features. Our goal is obtaining a rich library of *operation modules* and *open channels* to be used by the user, based on a deep study of the classical meta-heuristics algorithms for solving combinatorial problems, in order to cover them as much as possible. In such a way, building new algorithms by using POSL will be easier.

At the same time we pretend to develop new operators, depending on the new needs and requirements. It is necessary, for example, to improve the *solver definition* language, allowing the process to build sets of many new solvers to be faster

and easier. Furthermore, we are aiming to expand the communication definition language, in order to create versatile and more complex communication strategies, useful to study the solvers behavior.

As a medium term future work, we plan to include machine learning techniques, to allow solvers to change automatically, depending for instance on results of their neighbor solvers.

References

1. G. Blake, R.G. Dreslinski, T. Mudge, A survey of multicore processors. *IEEE Signal Process. Mag.* **26**(6), 26–37 (2009)
2. I. Boussaid, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics. *Inf. Sci.* **237**, 82–117 (2013)
3. A.E. Brownlee, J. Swan, E. Özcan, A.J. Parkes, Hyperion 2. A toolkit for {meta-, hyper-} heuristic research, in *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, GECCO Comp '14* (ACM, Vancouver, BC, 2014), pp. 1133–1140
4. S. Cahon, N. Melab, E.G. Talbi, ParadisEO: a framework for the reusable design of parallel and distributed metaheuristics. *J. Heuristics* **10**(3), 357–380 (2004)
5. D. Diaz, F. Richoux, P. Codognet, Y. Caniou, S. Abreu, Constraint-based local search for the costas array problem, in *Learning and Intelligent Optimization* (Springer, Berlin, Heidelberg, 2012), pp. 378–383
6. S. Frank, P. Hofstedt, P.R. Mai, Meta-S: a strategy-oriented meta-solver framework, in *Florida AI Research Society (FLAIRS) Conference* (2003), pp. 177–181
7. A.S. Fukunaga, Automated discovery of local search heuristics for satisfiability testing. *Evol. Comput.* **16**(1), 31–61 (2008)
8. M. Gendreau, J.Y. Potvin, Tabu search, in *Handbook of Metaheuristics*, vol. 146, 2nd edn. chap. 2, ed. by M. Gendreau, J.Y. Potvin (Springer, Berlin, 2010), pp. 41–59
9. D. Munera, D. Diaz, S. Abreu, P. Codognet, A parametric framework for cooperative parallel local search, in *Evolutionary Computation in Combinatorial Optimisation*, ed. by C. Blum, G. Ochoa. *Lecture Notes in Computer Science*, vol. 8600 (Springer, Berlin, Heidelberg, Granada, 2014), pp. 13–24
10. A.G. Nikolaev, S.H. Jacobson, Simulated annealing, in *Handbook of Metaheuristics*, vol. 146, 2nd edn., chap. 1, ed. by M. Gendreau, J.Y. Potvin (Springer, Berlin, 2010), pp. 1–39
11. B. Pajot, E. Monfroy, Separating search and strategy in solver cooperations, in *Perspectives of System Informatics* (Springer, Berlin, Heidelberg, 2003), pp. 401–414
12. A. Reyes-Amaro, E. Monfroy, F. Richoux, A parallel-oriented language for modeling constraint-based solvers, in *Proceedings of the 11th Edition of the Metaheuristics International Conference (MIC 2015)* (Springer, Berlin, 2015)
13. J. Swan, N. Burles, Templar - a framework for template-method hyper-heuristics, in *Genetic Programming*, ed. by P. Machado, M.I. Heywood, J. McDermott, M. Castelli, P. García-Sánchez, P. Burelli, S. Risi, K. Sim. *Lecture Notes in Computer Science*, vol. 9025 (Springer International Publishing, Cham, 2015), pp. 205–216
14. E.G. Talbi, Combining metaheuristics with mathematical programming, constraint programming and machine learning. *4OR* **11**(2), 101–150 (2013)

Chapter 7

An Extended Neighborhood Vision for Hill-Climbing Move Strategy Design

Sara Tari, Matthieu Basseur, and Adrien Goëffon

Abstract Many combinatorial optimization problem solvers are based on stochastic local search algorithms, which mainly differ by their move selection strategies, also called pivoting rules. In this chapter, we aim at determining pivoting rules that allow hill-climbing to reach good local optima. We propose here to use additional information provided by an extended neighborhood for an accurate selection of neighbors. In particular, we introduce the maximum expansion pivoting rule which consists in selecting a solution which maximizes the improvement possibilities at the next step. Empirical experiments on permutation-based problem instances indicate that the expansion score is a relevant criterion to attain good local optima.

Keywords Combinatorial optimization • Neighborhood search • Permutation problems

7.1 Introduction

Metaheuristics constitute a conceptual answer to tackle combinatorial problem instances that cannot be solved by complete methods using reasonable computational resources. In this work, we focus on neighborhood-based search techniques, which constitute a central component of most metaheuristics (e.g. simulating annealing, tabu search, iterated local search, memetic search) [20].

Neighborhood searches explore the search space by applying iteratively local modifications to a current solution thanks to a neighborhood relation. Strategies mainly differ by their selection criterion which determines the search trajectory and consequently the solutions reached.

In general, metaheuristic behaviors remain hard to analyze hence the difficulty to predict the comparative relevance of different selection criterions, also called *pivoting rule*. In order to enhance the understanding of neighborhood searches, and also to reduce the complexity of their behavior analysis, we focus here on

S. Tari • M. Basseur (✉) • A. Goëffon
LERIA, Université d'Angers, Angers, France
e-mail: sara.tari@univ-angers.fr; matthieu.basseur@univ-angers.fr;
adrien.goeffon@univ-angers.fr

hill-climbing techniques. In such strategies, only non-deteriorating moves are allowed to constitute possible trajectories. Traditionally, *best improvement* and *first improvement* strategies are two alternatives while designing a hill-climbing search.

In previous studies, we observed that best improvement is often preferred if the complete neighborhood evaluation does not increase significantly computational costs. However, we showed in [2, 5] that in many cases, best improvement is not the appropriate strategy to reach better solutions. We also investigated other pivoting rules as alternatives to best and first improvement. In particular, the behavior of the worst improvement was introduced in [3] to climb NK landscapes and presented in a more general context in the META 2014 conference [4].

As an extension of this work, this chapter focuses on determining advanced hill-climbing strategies which allow the attainment of higher local optima. Indeed, in [6], we pointed out that a hill-climbing process is often able to reach the best local optima provided that adequate moves are selected by the pivoting rule. Since meta-heuristic studies regularly emphasize that the behavior of their components strongly depends on the search space structure considered, we choose here to focus the study on two permutation problems: Flowshop and QAP.

The advanced pivoting rules investigated in this chapter consider an extended neighborhood as additional information for choosing the move within the neighborhood originally defined. We are mainly interested on the *maximum expansion* strategy, which consists in moving towards solutions maximizing the number of move possibilities for the next iteration. We include to experimental comparisons several hill-climbing strategies which use or not such an extended vision of the search space.

In the next section, we recall main notions and definitions related to combinatorial optimization and local search. Section 7.3 describes hill-climbing variants as well as experimental comparisons. In the light of these results, we propose in Sect. 7.4 possible ways to enhance search processes by multiobjectivization. We finally conclude by a discussion and point out some perspectives.

7.2 Combinatorial Optimization, Local Search, Hill-Climbing

A combinatorial instance problem can be defined as a pair (\mathcal{X}, f) , where \mathcal{X} is a discrete set of feasible solutions called *search space*, and $f : \mathcal{X} \rightarrow \mathbb{R}$, a scalar *objective function* which has to be maximized or minimized. Solving an optimization problem (\mathcal{X}, f) consists in finding $x^* \in \operatorname{argmax}_{x \in \mathcal{X}} f(x)$. Note that here f has to be maximized, but minimization problems can be considered without loss of generality.

A local search algorithm (Algorithm 1) consists in navigating through the search space thanks to a neighborhood function $\mathcal{N} : \mathcal{X} \rightarrow 2^{\mathcal{X}}$ which assigns to each $x \in \mathcal{X}$ a set of neighboring solutions, and an *evaluation function* which allows a partial order relation between solutions. In the following, we will use the objective

function f as evaluation function. A solution $x' \in \mathcal{N}$ is a *neighbor* of x . If $f(x') > f(x)$ (resp. $=$, $<$), then the transition $x \rightarrow x'$ is an improving (resp. neutral, deteriorating) move. x' is then an improving (resp. neutral, deteriorating) neighbor. A *global optimum* is a solution $x^* \in \operatorname{argmax}_{x \in \mathcal{X}} f(x)$, i.e. an optimal solution of the combinatorial problem (\mathcal{X}, f) . A *local optimum* is a solution x such that $\forall x' \in \mathcal{N}(x), f(x') \leq f(x)$. A *strict local optimum* have only deteriorating neighbors. We call *neutral perturbation* a neutral move $x \rightarrow x'$ such that x is a local optimum.

Given (\mathcal{X}, f) and \mathcal{N} , the quality of a local search algorithm resides in its ability to reach good solutions (thanks to the pivoting rule used, exploiting f) in reasonable computational costs. A local search process can be seen as a particular sampling of \mathcal{X} using \mathcal{N} and f . Triplets $(\mathcal{X}, \mathcal{N}, f)$ are called *fitness landscapes* [10] when abstracted from problem-oriented issues. Their particular analysis sheds light on possible links between structural properties and local searches behavior.

Algorithm 1 Local search

```

Choose  $x \in \mathcal{X}$  (initialization)
 $x_b \leftarrow x$  (save the best solution found)
repeat
  Choose  $x' \in \mathcal{N}(x)$  w.r.t. a piv. rule
   $x \leftarrow x'$ 
  if  $f(x) > f(x_b)$  then
     $x_b \leftarrow x$ 
  end if
until stop criterion
Return  $x_b$ 

```

Algorithm 2 Hill-climbing

```

Choose  $x \in \mathcal{X}$  (initialization)
repeat
   $\mathcal{N}_+(x) = \{y \in \mathcal{N}(x), f(y) > f(x)\}$ 
  if  $\mathcal{N}_+(x) \neq \emptyset$  then
    Choose  $x' \in \mathcal{N}_+(x)$  w.r.t. a piv. rule
     $x \leftarrow x'$ 
  end if
until  $\mathcal{N}_+(x) = \emptyset$  /*  $x$  is a local opt. */
Return  $x$ 

```

A local search algorithm is called *hill-climbing*, or *climber*, if the pivoting rule does not allow deteriorating moves (see Algorithm 2). Thus, it is not necessary to save the best solution encountered during the search (x_b in Algorithm 1). A *strict hill-climbing* allows only improving moves, while a *stochastic hill-climbing* allows also neutral moves. If no neutral move is permitted during the search (including

neutral perturbations), then reaching a local optimum is a natural stop criterion. Otherwise, the search stops when a strict local optimum is reached, or if a maximal number of iterations or evaluated solutions is attained.

A *best improvement* strict hill-climbing selects at each iteration the best improving neighbor, while a (random) *first improvement* strict hill-climbing selects any improving neighbor. Note that the entire neighborhood of a current solution has not to be evaluated in a first improvement climber. Different pivoting rules can be defined by combining first or best improvement strategy with a stochastic hill-climbing process. Other strategies will be discussed in the next section.

The difficulty of solving a problem (\mathcal{X}, f) with a local search using a neighborhood \mathcal{N} greatly depends on the characteristics of its associated fitness landscape [14]. In particular, the ruggedness of $(\mathcal{X}, \mathcal{N}, f)$ express its amount of epistasis phenomenon [7] and the number of local optima which constitute the main obstacle to climbers (see [17, 5] for more details). *Neutrality*, which refers to the amount of neutral transitions and plateaus, affects also the capacity of local searches to explore landscapes efficiently.

Previous works [5] showed that some hill-climbing pivoting rules are generally more appropriate and their efficiency could differ according to landscape properties. In particular, when the landscape is sufficiently rugged, a best improvement strategy leads often prematurely to local optima, while pivoting rules which favor small improvements lead to longer searches which often drive to better solutions. Based on these observations, it seems relevant to develop pivoting rules dedicated to preserve as much as possible improvement options. This leads us to propose the *maximum expansion* strategy which uses information of additional degree of neighborhood. The next section is dedicated on maximum expansion which is experimentally compared with other climbing strategies.

7.3 Hill-Climbing Moving Strategies: Description and Evaluation

7.3.1 Context

In this work, we investigate specific pivoting rules which aim to enhance hill-climbing performance by exploiting the knowledge of a wider area than the considered neighborhood. We call *k-th level neighborhood* of a configuration x the set of solutions $\mathcal{A}_k(x) = \{x' \in \mathcal{X}, d_{\mathcal{N}}(x, x') \leq k\}$, where $d_{\mathcal{N}}(x, x')$ refers to the minimal number of moves to link x to x' with respect to \mathcal{N} . As a consequence, $\mathcal{A}_1(x) = \mathcal{N}(x) \cup \{x\}$. In the following, we will distinguish (1) the *moving area* and (2) the *vision area* of a local search algorithm. The moving area is the set of solutions reachable in a single step of the search (independently of any selection criterion). The vision area is composed of solutions which may be used by the pivoting rule to choose a solution from the moving area (see Fig. 7.1).

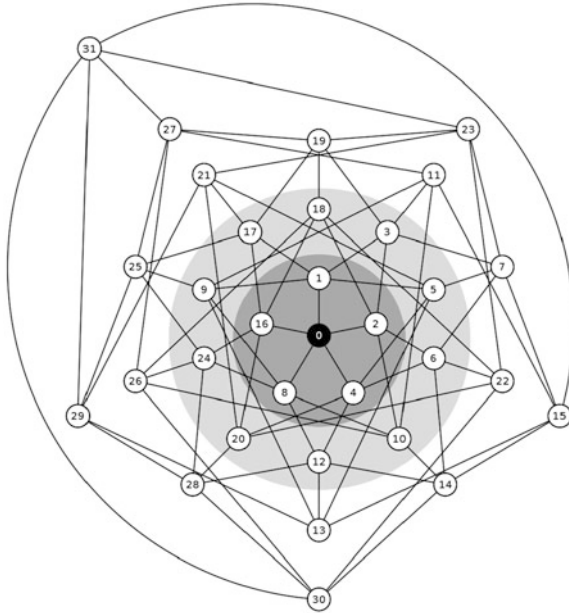


Fig. 7.1 Hypercube of dimension 5 where vertices are placed in order to respect the relation between hamming distance and Euclidian distance to the solution “00000”. We aim at moving efficiently in the *moving area* thanks to the information provided by the *vision area*

Classical pivoting rules such as first and best improvement use the first level neighborhood \mathcal{A}_1 as vision area. At each step of the search, this vision area provides only information of the current solution and its neighbors. The *maximum expansion* hill-climbing pivoting rule ($ME_{>}$), which selects the improving neighbor maximizing its own number of improving neighbors, uses an extended vision area (\mathcal{A}_2). Let us notice that such a pivoting rule has been considered in a complexity study for the p-median problem [1].

This section is dedicated to assess the efficiency of $ME_{>}$ in comparison to other hill-climbing strategies. In a first study (Sect. 7.3.2) $ME_{>}$ is compared to first level hill-climbing strategies (best, first and worst improvement). Climbers based on vision area \mathcal{A}_2 , including $ME_{>}$, are presented and competed in a specific study (Sect. 7.3.3).

7.3.2 Experimental Protocol

Despite the generic aspects of metaheuristics, their efficiency generally depends on problem specificities. In particular, different solution representations (bit strings, permutations, assignments) lead to different search space structures. Through the

study of NK landscapes and MAXSAT problems [5], we already have an effective understanding of the behavior of local searches on bit string landscapes. Here we focus on permutation-based problems. Experiments are conducted on Flow-shop and QAP instances.

The *Flow-shop Scheduling Problem* (FSP) [19] consists in scheduling n jobs on m machines. A machine cannot be assigned to two jobs simultaneously. Each job is composed of m consecutive tasks with specific processing times. In the permutation variant under consideration, jobs must be scheduled in the same order on all machines, and the objective value to be minimized is the total completion time called *makespan* (see [5] for a more formal description). Note that this problem has been proved to be NP-hard for more than two machines [13].

The *Quadratic Assignment Problem* (QAP) [12] is an other NP-hard permutation problem [18, 16] which aims at assigning a set of n facilities to a set of n locations with given distances between locations and given flows between facilities. The objective value to be minimized corresponds to the sum of the products between flows and distances, relatively to a permutation describing the assignment.

FSP and QAP are both permutation problems and thus involve common search spaces. However, neighborhoods traditionally used for tackling these two problems differ (*insertion* for FSP and *swap* for QAP). Then two main features distinguish instance characteristics: the problem under consideration and the permutation size. 117 Flow-shop and 93 QAP instances, with various size and ruggedness level, were used for experimentations.¹

The experimental comparison of hill-climbing strategies follows a specific experimental protocol. For each couple (instance, climber), 100 executions are performed from identical sets of 100 randomly generated solutions, in order to reduce stochastic bias. Two criteria are used for comparing two strategies A and B: the global success ratio² of A against B from identical starting solutions, and the ratio of instances where A statistically dominates B. The statistical dominance of a strategy over an other is assessed from the number of successes on a particular instance with respect to a binomial test (p -value 0.95). More precisely, if \mathcal{S} denotes the number of successes of method A over method B after 100 confrontations, then A statistically outperforms B when $\frac{1}{2^{100}} \sum_{i=0}^{\mathcal{S}} \binom{100}{i} \geq 0.95$, i.e. $\mathcal{S} \geq 58$.

7.3.3 Maximum Expansion vs. \mathcal{A}_1 Vision Area Climbers

The first empirical comparison proposed in this section focuses on competing ME_> with three basic hill-climbing strategies. The four variants under consideration are the following ones:

¹ FSP instances are taken from [19]. Their sizes vary from 20 to 50 jobs and 5 to 20 machines. QAP Instances can be found on opt.math.tu-graz.ac.at/qaplib/inst.html. Instances used for tests involve permutations of size ranging from 12 to 64.

² We call *success* the event 'A reaches a strictly better solution than B from the same starting solution'.

- BEST selects at each step the best improving neighbor.
- FIRST selects at each step an improving neighbor.
- WORST selects at each step the least improving neighbor.
- ME_> selects at each step the improving neighbor with the best *expansion score*. The expansion score of a solution denotes its number of improving neighbors. If the best expansion score is equal to 0, meaning that all improving neighbors are local optima, then ME_> selects the best one thanks to f (see Algorithm 3).

Algorithm 3 ME_> hill-climbing

```

1: Choose  $x \in \mathcal{X}$  (initialization)
2:  $N_+(x) = \{y \in \mathcal{N}(x), f(y) > f(x)\}$ 
3: for each  $y_i \in N_+(x)$  do
4:    $E(y_i) = \#\{z \in N(y_i), f(z) > f(y_i)\}$ 
5: end for
6: if  $\max_i E(y_i) > 0$  then
7:    $x \leftarrow x_j, j \in \text{argmax}_i E(y_i)$ 
8:   Go to 2
9: else
10:   $x \leftarrow x_j, j \in \text{argmax}_f N_+(x)$ 
11: end if
12: Return  $x$ 

```

When several neighbors satisfy the moving criterion, one of them is chosen randomly. Algorithms stop when a local optimum is attained. Let us recall that all these variants never accept neutral or deteriorating moves. Figure 7.2 depicts the three most determinist strategies.

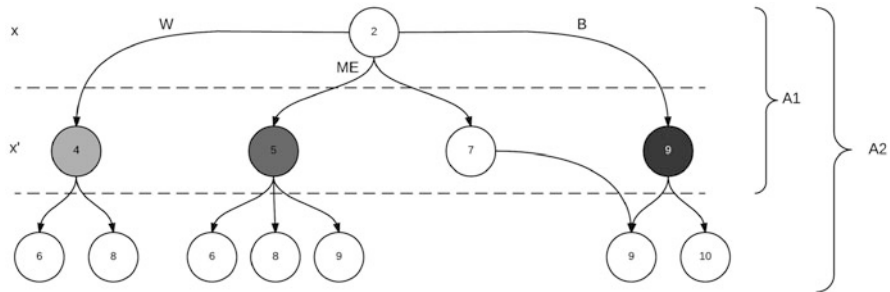






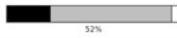

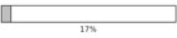
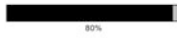

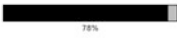


Fig. 7.2 Illustration of climber variants behavior starting from solution x . Only improving moves are represented









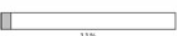



Naturally these different pivoting rules involve a different behavior during the search. Tables 7.1 and 7.2 summarize the experiments realized through performance features described previously and allow the visualization of their relative capacity to reach high local optima.

Table 7.1 Pairwise comparisons of pivoting rules on QAP instances

	F	B	W	ME
F		 47%	 43%	 14%
B	 47%		 43%	 14%
W	 52%	 52%		 17%
ME	 80%	 80%	 78%	

Bars indicate proportions of instances leading to the three possible statistical conclusions. Black areas represent the number of instances where strategy A (in line) statistically dominates strategy B (in column), w.r.t. a binomial test (with a p -value of 0.95). Grey areas represent instances where no statistical conclusion is obtained. White areas represent instances where strategy A is statistically dominated by strategy B. Finally, the value below each bar corresponds to the proportion of successes considering all executions on all instances

Table 7.2 Pairwise comparisons of pivoting rules on FSP instances

	F	B	W	ME
F		 52%	 41%	 10%
B	 47%		 35%	 8%
W	 52%	 52%		 11%
ME	 84%	 86%	 83%	

The main result here concerns $ME_{>}$, which clearly outperforms the 3 other methods. Indeed, $ME_{>}$ statistically dominates $WORST$, $FIRST$ and $BEST$ on more than 90% of instances and is never statistically dominated. In the light of this observation, we assume that the efficiency of climbers is directly induced by the expansion score of the solutions selected during the search.

Let us recall that we focus here on the capacity of a hill-climbing pivoting rule to reach local optima as highest as possible. In terms of computational costs, $ME_{>}$ is clearly slower than other variants since it considers the second neighborhood level at each step of the search.

Experiments show that generally $WORST$ is better than classical $BEST$ and $FIRST$ climbers on the considered permutation instances, and more particularly on FSP instances (see Tables 7.1 and 7.2). $BEST$ is more efficient than $WORST$ on some QAP instances with specific properties, but globally $FIRST$ often leads to better local optima than $BEST$. This confirms the general observation $(FIRST \succ BEST) \Leftrightarrow (WORST \succ FIRST)$ [3].

Since $WORST$ outperforms the other first level pivoting rules studied, we wonder if its relative efficiency comes from its tendency to select neighbors with a high expansion score. To emphasize a possible link between $WORST$ and $ME_{>}$ behaviors, we collected average ranks of selected solutions among improving neighbors as follow.

Let $r = |\{x'' \in \mathcal{N}_+(x), f(x'') > f(x')\}| + \frac{1}{2}|\{x'' \in \mathcal{N}_+x, f(x'') > f(x')\}|$ the rank of x' within the set of improving neighbors $\mathcal{N}_+(x)$ of size n . Provided that k is the number of ranking classes, x' belong to a class C_i with a ratio $\max(0, \min(r/n, i/k) - \max(r - 1/n, i - 1/n))/n$.

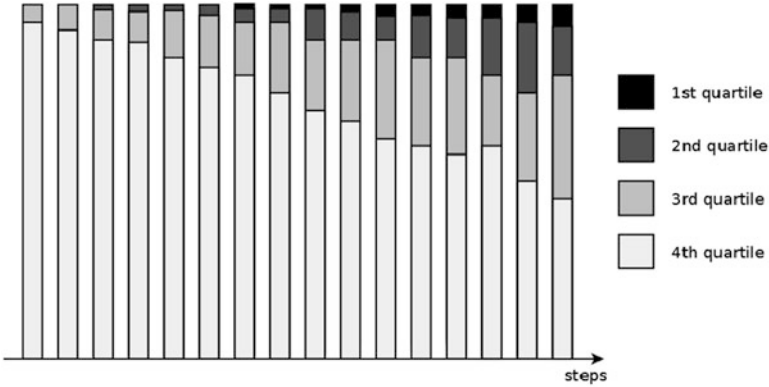


Fig. 7.3 Evolution of rank classes of selected neighbor during $ME_{>}$ processes

Figure 7.3 reports the average values of C_i ratios considering $k = 4$ classes. One can observe that most of the selected neighbors of $ME_{>}$ belong in the class C_4 , which is constituted of the 25% least improving neighbors. It indicates that there exists a certain level of similitude between $ME_{>}$ and WORST behaviors.

Although $ME_{>}$ clearly outperforms the first level pivoting rules, it takes advantage of additional information provided by its extended vision. To achieve a more accurate evaluation of the expansion score criterion relevance, we proceeded to a second study which includes other second level pivoting rules.

7.3.4 Maximum Expansion vs. \mathcal{A}_2 Vision Area Climbers

We note \mathcal{C}_m^v a climber based on pivoting rule \mathcal{C} , using \mathcal{A}_v as vision area and \mathcal{A}_m as moving area. In this section, we introduce additional climbers involving the second level neighborhood as vision area ($v = 2$). Pivoting rules best (B) and first (F) improvement are used to provide two variants of extended vision hill-climblings: climbers whose moves are restricted to the original neighborhood \mathcal{N} ($m = 1$) and climbers which use the large neighborhood \mathcal{A}_2 . This leads to the following variants:

- B_1^2 and F_1^2 use the same moving and vision areas as $ME_{>}$. At each step of the search, B_1^2 starts by identifying the best solution x_b among the vision area. x_b is selected if it belongs to the moving area, otherwise B_1^2 selects the improving neighboring solution which leads to x_b . The first improvement variant is similar to the basic climber FIRST, except that no local optimum can be selected if other alternatives exist. Despite B_1^2 and F_1^2 are directly derived from classical pivoting

rules, some local optima can be avoided by using \mathcal{A}_2 as vision area. Let us notice that B_1^2 corresponds to the *two steps hill-climbing* proposed in [8].

- B_2^2 and F_2^2 represent best and first strict improvement hill-climbings employing the large neighborhood $\mathcal{N}' = \mathcal{A}_2$, commonly denoted as $\mathcal{N} \cup \mathcal{N}^2$. Note that B_2^2 and F_2^2 allow more search paths than all other variants studied (including $ME_{>}$) since they involve a larger neighborhood. Although these strategies remain climbers (with respect to their extended neighborhood), they navigate through a search space having less local optima than when considering neighborhood \mathcal{N} .

Worst improvement variants W_1^2 and W_2^2 were also considered for experiments, but related tests were dropped on account of too high computation times.

In a first set of experiments, we compete the maximum expansion strategy $ME_{>}$ against B_1^2 and F_1^2 . Figures 7.4 and 7.5 reports results of pairwise comparisons between these three strategies. Note that here all climbers use the same moving and vision areas ($m = 1, v = 2$), which constitutes the fairest comparison for evaluating $ME_{>}$. One can observe that both B_1^2 and F_1^2 are largely dominated by $ME_{>}$ despite the use of \mathcal{A}_2 . There is neither QAP nor FSP instance where $ME_{>}$ is statistically dominated here. These results show that the maximum expansion pivoting rule efficiency cannot only be explained by its immediate capacity to avoid local optima, common to all variants sharing the knowledge of a larger neighborhood. Globally, this comparison indicates that the expansion score is a relevant criterion for guiding the search to good local optima.

The comparison of $ME_{>}$ with other \mathcal{C}_1^v climbers showed that maximum expansion is a relevant pivoting rule for hill-climbing. In order to measure its efficiency in a more general way, we propose to compare $ME_{>}$ with less restrictive strategies which

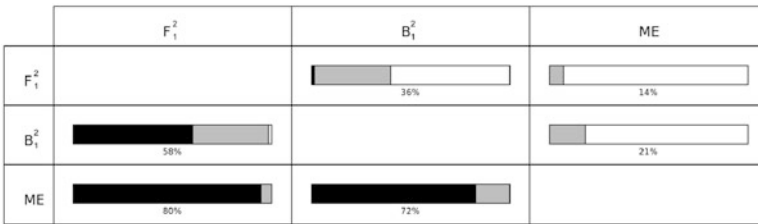


Fig. 7.4 Pairwise comparisons of pivoting rules on QAP instances: $ME_{>}$ vs. first level pivoting rules

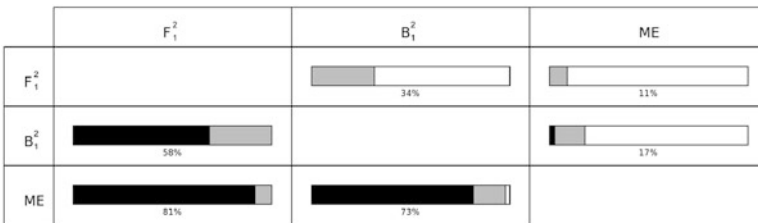


Fig. 7.5 Pairwise comparisons of pivoting rules on FSP instances: $ME_{>}$ vs. first level pivoting rules

allow the selection of improving neighbors belonging to the second level neighborhood. The aforementioned B_2^2 and F_2^2 are large neighborhood hill-climbings which use the vision area of $ME_{>}$ as moving area. Figures 7.6 and 7.7 report comparison results proceeded in this second set of experiments. We note that the global efficiencies of $ME_{>}$, B_2^2 and F_2^2 are relatively homogeneous. This observation indicates that even with the constraint of choosing solutions within a smaller neighborhood, $ME_{>}$ reaches equivalent local optima than large neighborhood hill-climbings. Thus, it is possible to define hill-climbing pivoting rules able to bypass the barrier of low-quality local optima usually encountered by traditional climbers. A potential issue resulting from this analysis is how to reduce the quantity of knowledge (here the vision area) while maintaining a similar level of performance.

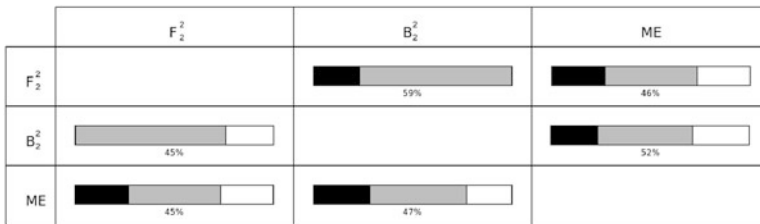


Fig. 7.6 Pairwise comparisons of pivoting rules on QAP instances: $ME_{>}$ vs. second level pivoting rules

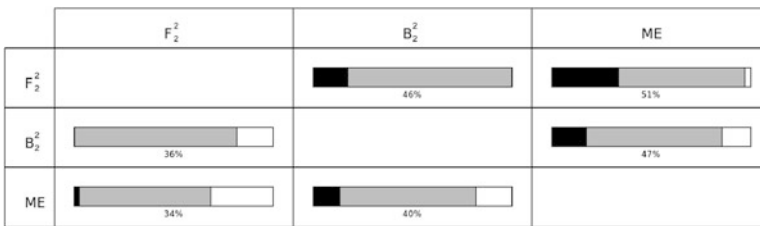


Fig. 7.7 Pairwise comparisons of pivoting rules on FSP instances: $ME_{>}$ vs. second level pivoting rules

7.4 Maximum Expansion Sophistication: A Multiobjectivized Approach

Expansion score has been shown to be a pertinent indicator for guiding the search towards high local optima on landscapes derived from permutation problems. Naturally, a local search algorithm which uses exclusively this selection criterion could not be able to reach good solutions since deteriorating moves will be mostly performed. As hill-climbing requires to select non-deteriorating neighbors only, the evaluation score constitutes then an essential information which determines the

behavior of every climber, including $\text{ME}_{>}$. Intuitively, it seems interesting to consider tradeoffs between reaching good solutions and preserving their expansion score. In this section, we propose multiobjectivized approaches to define climbers handling expansion and evaluation functions. In the following, we first introduce multiobjectivization and provide briefly a few definitions. Then we propose biobjectivized pivoting rules. These pivoting rules are then evaluated as in the previous section.

7.4.1 Multiobjectivization

The multiobjectivization of a single objective optimization problem (\mathcal{X}, f) is to consider a multiobjective problem $(\mathcal{X}, (f_1, \dots, f_n))$ in order to ease the resolution of (\mathcal{X}, f) [11]. Multiobjectivizing a problem can be achieved either by adding objective functions to the original one f , or by replacing f with a set of new objectives. We propose to adapt the multiobjectivization principle, by using pivoting rules which involve a biobjective evaluation function F for climbing single objective landscapes.

Here, we use a function $F = (f_1, f_2)$ which considers evaluation and expansion scores such that:

$$\begin{cases} f_1(x) = f(x) & \text{(original evaluation score)} \\ f_2(x) = \#\{x' \in \mathcal{N}(x), f(x') > f(x)\} & \text{(expansion score)} \end{cases}$$

Note that even if it is not required by the principle of multiobjectivization, here f is a component of F ($f_1 = f$).

In multiobjective optimization, we usually seek for a set of solutions which offer good compromises of the objective functions. Recall that there exists only a partial order relation between solutions (*Pareto dominance*). A solution x_i is said to *dominate* x_j with respect to F ($x_i \succeq_F x_j$) if and only if:

$$\begin{cases} \forall k \in [1..n], f_k(x_i) \geq f_k(x_j) \\ \exists k \in [1..n] \text{ s.t. } f_k(x_i) > f_k(x_j) \end{cases}$$

A solution x is non-dominated by a set of solutions S with respect to F if and only if $\forall y \in S, y \not\succeq_F x$. We note $x \not\prec S$.

Here, we still focus on climbing single-objective landscapes (thanks to the original evaluation function f), but with pivoting rules which involve the biobjective evaluation function F . We next propose pivoting rules which consider only moves $x \rightarrow x'$ satisfying the two following constraints: (1) $f(x') > f(x)$, and (2) x' is not dominated by $\mathcal{N}(x)$ w.r.t. F .

7.4.2 Biobjectivized Pivoting Rules

We defined three biobjectivized pivoting rules (BO_*) in order to observe the effect of different compromises between improving the solution evaluation and maximizing the expansion score.

Let $\mathcal{N}_>(x) = \{x' \in \mathcal{N}(x), f(x') > f(x)\}$ be the set of strictly improving neighbors of a solution x w.r.t. an evaluation function f . We note $\mathcal{N}_{(>,\geq)}(x) = \{x' \in \mathcal{N}_>(x), x' \not\leq \mathcal{N}_>(x)\}$ the set of non-dominated improving neighbors of x w.r.t. f and a multiobjectivized function F .

At each step of the search, the three climbers proposed in this section consists in choosing a solution within the restricted neighborhood $\mathcal{N}_{(>,\geq)}(x)$ of a solution x . These variants work as follow (see also Fig. 7.8):

- BO_{rand} selects randomly a solution within $\mathcal{N}_{(>,\geq)}(x)$.
- $\text{BO}_{\text{H(exp)}}$ selects the solution which maximizes the evaluation score among the top half solutions of $\mathcal{N}_>(x)$ of higher expansion score.
- $\text{BO}_{\text{H(eval)}}$ selects the solution which maximizes the expansion score among the top half solutions of $\mathcal{N}_>(x)$ of higher evaluation score.

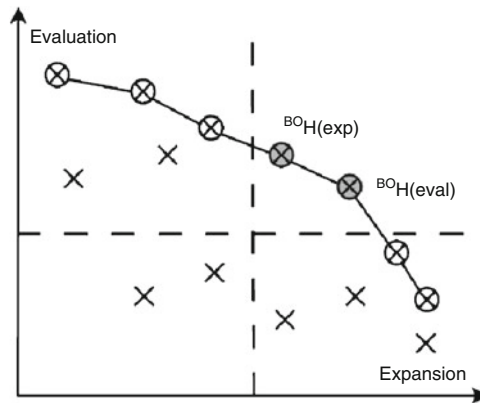


Fig. 7.8 Biobjectivized pivoting rules. $\text{BO}_{\text{H(exp)}}$ and $\text{BO}_{\text{H(eval)}}$ selection strategies are deterministic, whereas BO_{rand} selects randomly a non-dominated solution (circled crosses)

Figure 7.9 reports the analysis of the empirical comparison realized on QAP instances, considering climbers with biobjectivized pivoting rules as well as $\text{ME}_>$. BO_{rand} appears to be clearly outperformed by other variants.

Although there is no statistical difference between $\text{ME}_>$ and both $\text{BO}_{\text{H}(\ast)}$ climbers on a significant number of instances, $\text{ME}_>$ is almost never dominated by any other variant. Among the three biobjectivized climbers experimented, $\text{BO}_{\text{H(eval)}}$, which is the most efficient one, is not frequently dominated by $\text{ME}_>$. Recall that $\text{BO}_{\text{H(eval)}}$ restricts the possible selected neighbors according to their evaluation score, but the expansion score is at least the criterion being maximized. These observations lead us

to believe that the expansion score is the only criterion which directly affects the overall capacity of a climber to reach high local optima. Thus, BEST and WORST efficiencies would ensue from their respective ability to select solutions with high expansion scores.

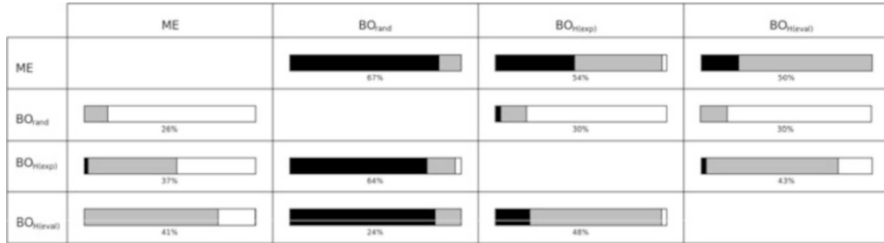


Fig. 7.9 Pairwise comparisons of pivoting rules on QAP instances: $ME_{>}$ vs. biobjectivized second level pivoting rules

7.5 Discussion

In this chapter, we were interested in studying the ability of different climber variants to reach good local optima. In particular, we investigated how to take advantage of the use of a second level neighborhood for deciding which move to select in the first level neighborhood. This led us to propose the maximum expansion strategy (ME) which consists in moving towards solutions maximizing the number of improvement possibilities. The comparison of ME with other pivoting rules emphasizes its effectiveness. Notably, ME is often competitive against large neighborhood climbers which have the advantage of working on landscapes containing fewer local optima.

Previous studies already compared the efficiency of classical (first level) pivoting rules on specific problems [9, 15, 21, 2] and pointed out that the relative efficiency of climbers can be strongly dependent of instances characteristics, e.g. their associated landscape size, ruggedness and neutrality. In particular, first and worst improvement seems more adapted for climbing Flowshop landscapes, whereas climber efficiency is more instance-dependent while considering QAP. ME is efficient on permutation instances and does not seem to be significantly affected by their characteristics, despite we pointed out some similarities between ME and WORST behaviors. More experiments should be realized on a large scale of various optimization problem instances to corroborate these aspects.

Let us recall that our aim was here to determine ways to reach good local optima without considering computational costs issues. Obviously, ME is more time-consuming than classical climbers which do not require the knowledge of

an extended neighborhood. An interesting perspective consists in defining pivoting rules that approximate the behavior of ME in a reduced computational effort.

Finally, ongoing work includes the extension to expansion-based local searches by investigating ways to design less restrictive pivoting rules. Since the incorporation of neutral moves within hill-climbing strategies leads to more efficient searches, we experimented a neutral version of ME. It appears that such a strategy is able to outperform other neutral versions of climbers. The ways to extend ME to other types of local searches (i.e. allowing deteriorating moves) still can be improved since it brings out some difficulties, notably cycling issues. Nevertheless, advanced metaheuristics could be enhanced by considering the expansion criterion in their neighborhood search components.

Acknowledgements The work is partially supported by the PGMO project from the *Fondation Math'ematique Jacques Hadamard*.

References

1. E. Alekseeva, Y. Kochetov, A. Plyasunov, Complexity of local search for the p-median problem, in *18th Mini Euro Conference on VNS*, 2005
2. M. Basseur, A. Goëffon, Hill-climbing strategies on various landscapes: an empirical comparison, in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, GECCO '13* (ACM, New York, 2013), pp. 479–486
3. M. Basseur, A. Goëffon, On the efficiency of worst improvement for climbing NK-landscapes, in *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation, GECCO '14* (ACM, New York, 2014), pp. 413–420
4. M. Basseur, A. Goëffon, Unconventional pivoting rules for local search, in *International Conference on Metaheuristics and Nature Inspired Computing (META'2014)*, 2014
5. M. Basseur, A. Goëffon, Climbing combinatorial fitness landscapes. *Appl. Soft Comput.* **30**, 688–704 (2015)
6. M. Basseur, A. Goëffon, F. Lardeux, F. Saubion, V. Vigneron, On the attainability of NK landscapes global optima, in *Seventh Annual Symposium on Combinatorial Search*, 2014
7. W. Bateson, *Mendel's Principles of Heredity* (Cambridge University Press, Cambridge, 1909)
8. P. Collard, S. Verel, M. Clergue, How to use the scuba diving metaphor to solve problem with neutrality? in *ECAI 2004: 16th European Conference on Artificial Intelligence, 22–27 August 2004, Valencia, Spain: Including Prestigious Applications of Intelligent Systems (PAIS 2004): Proceedings*, vol. 110 (IOS Press, Amsterdam, 2004), p. 166
9. P. Hansen, N. Mladenovic, First vs. best improvement: an empirical study. *Discret. Appl. Math.* **154**(5), 802–817 (2006). {IV} ALIO/EURO Workshop on Applied Combinatorial Optimization IV ALIO/EURO Workshop on Applied Combinatorial Optimization
10. S.A. Kauffman, *The Origins of Order: Self-organization and Selection in Evolution* (Oxford University Press, New York, 1993)
11. J.D. Knowles, R.A. Watson, D. Corne, Reducing local optima in single-objective problems by multi-objectivization, in *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, Zurich, 7–9 March 2001, pp. 269–283, 2001
12. T. Koopmans, M.J. Beckmann, Assignment problems and the location of economic activities. Cowles Foundation Discussion Papers 4, Cowles Foundation for Research in Economics, Yale University, 1955

13. J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems. *Ann. Discret. Math.* **1**, 343–362 (1977)
14. K. Malan, A.P. Engelbrecht, A survey of techniques for characterising fitness landscapes and some possible ways forward. *Inf. Sci.* **241**, 148–163 (2013)
15. G. Ochoa, S. Verel, M. Tomassini, First-improvement vs. best-improvement local optima networks of NK landscapes, in *Parallel Problem Solving from Nature, PPSN XI*, ed. by R. Schaefer, C. Cotta, J. Koodziej, G. Rudolph. Lecture Notes in Computer Science (Springer, Berlin/Heidelberg, 2010), pp. 104–113
16. P.P.M. Pardalos, H. Wolkowicz, *Quadratic Assignment and Related Problems: Dimacs Workshop 20–21 May 1993*, vol. 16 (American Mathematical Society, Providence, RI, 1994)
17. F.J. Poelwijk, S. Tanase-Nicola, D.J. Kiviet, S.J. Tans, Reciprocal sign epistasis is a necessary condition for multi-peaked fitness landscapes. *J. Theor. Biol.* **272**(1), 141–144 (2011)
18. S. Sahni, T. Gonzalez, P-complete approximation problems. *J. ACM* **23**(3), 555–565 (1976)
19. É. Taillard, Benchmarks for basic scheduling problems. *Eur. J. Oper. Res.* **64**(2), 278–285 (1993)
20. E.-G. Talbi, *Metaheuristics: From Design to Implementation*, vol. 74 (Wiley, Hoboken, 2009)
21. D. Whitley, A.E. Howe, D. Hains, Greedy or not? Best improving versus first improving stochastic local search for MAXSAT, in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 14–18 July 2013, Bellevue, Washington (2013)

Chapter 8

Theory Driven Design of Efficient Genetic Algorithms for a Classical Graph Problem

Dogan Corus and Per Kristian Lehre

Abstract This paper presents a principled way of designing a genetic algorithm which can guarantee a rigorously proven upper bound on its optimization time. The shortest path problem is selected to demonstrate how level-based analysis, a general purpose analytical tool, can be used as a design guide. We show that level-based analysis can also ease the experimental burden of finding appropriate parameter settings. Apart from providing an example of theory-driven algorithmic design, we also provide the first runtime analysis of a non-elitist population-based evolutionary algorithm for both the single-source and all-pairs shortest path problems.

Keywords Runtime analysis • Genetic algorithms • Level-based analysis • Shortest path problems

8.1 Introduction

Evolutionary algorithms (EAs) have been a popular class of heuristic optimization techniques since the 1960s. They are inspired by natural evolution where limited resources allow only the fittest individuals to reproduce and drive progress over generations. Evolutionary algorithms generate new solutions from existing ones with small random changes and select solutions with higher fitness function values to survive in the next generation. The generality of the described process makes EAs general purpose methods which can be applied to any optimization problem defined on any search space. All that is required is a way to represent candidate solutions as individuals of the population (i.e. the representation) and some measure of solution quality (i.e. the fitness function). As a result, EAs do not require any information

D. Corus (✉)
The University of Sheffield, Sheffield S10 2TN, UK
e-mail: d.corus@sheffield.ac.uk

P.K. Lehre
School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK
e-mail: P.K.Lehre@bham.ac.uk

about the problem structure apart from the fitness value of candidate solutions (i.e. they are black-box optimization algorithms [13]). Although they are widely used by practitioners [14], the theory behind how EAs work was not established until much later. While the scope of early theoretical work was limited to toy problems and simple algorithms, the gap between what practitioners and theoreticians are interested in gradually closes.

Runtime analysis of evolutionary algorithms has been a rapidly growing research field for almost two decades [21, 15, 1]. The shortest path problems is one of the first examples of the runtime analysis of EAs on a practically relevant problem. For practitioners, it is common to use variation and selection operators which are tailored to the problem at hand. Still, previous works on runtime analysis of EAs seldom digressed from elementary operators. Because, first of all, the tailored operators both complicated the analysis and undermined the generality of the runtime results. Even negative results for elementary algorithms, i.e. that the elementary algorithm requires exponential time to reach a desired solution quality, served the purpose of pointing out a potential weakness of EAs in general. Moreover, early runtime analysis results considered problems which were either too simple or too artificial to justify tailoring operators to fit the problem. When a negative result was proven for a non-artificial combinatorial optimization problem like the shortest path problem [22], then designing the simplest EA which can solve it efficiently became admissible. A series of works provided positive results by adapting the problem representation, the objective function, the variation and selection operators and adding auxiliary mechanisms that improve solution quality. Eventually, the requirement for more complex operators made the all-pairs shortest path problem (APSP) the first non-artificial problem where an EA with crossover is proven to perform asymptotically better than an EA without [10].

Extending the scope of runtime analysis to non-artificial problems motivated the analyzes of more complex EAs but did not provide the means to do such analysis. Introduction of new analytical tools like drift analysis that allow obtaining results that are otherwise beyond the reach of older approaches responded to this need. A new tool for runtime analysis of evolutionary algorithms that builds on top of drift analysis is the level-based method proposed in [3]. This general tool covers a wide range of problems and algorithms including those with a complex solution space and complex operators. The theorem provides conditions that are sufficient to guarantee a polynomial runtime, and often these conditions are also necessary. The necessary conditions include a sufficient selective pressure, upgrade probability and population size. While it might be hard to determine whether an existing algorithm satisfies these conditions, it is easier to design an algorithm that does satisfy the conditions. In this paper we will use the well known shortest path problem to demonstrate how to construct an algorithm guided by the conditions of this theorem. We will go through how the design choices allow us to rigorously prove the performance with a very brief and simple analysis at the end. By doing that, we will also provide the first runtime analysis of a non-elitist genetic algorithm for the single-source shortest path problem and all-pairs shortest path problem.

This paper is structured as follows: In the following chapter the shortest path problem will be formally defined and relevant research will be summarized. Level-based analysis will be briefly introduced in Sect. 8.3. In Sect. 8.4, the algorithm will be explained component by component and the rationale behind the selection of components will be related to the conditions of the level-based theorem. At the end of Sect. 8.4, the runtime will be proven using a corollary to the level-based theorem and the last section will draw the conclusions.

Throughout this paper we will use the notation $[n]$ to represent $\{1, 2, \dots, n\}$, the set of natural numbers up to n and the notation $X \sim D$ which denotes that X is a random element sampled from the distribution D .

8.2 Analysis of EAs on Shortest Path Problems

An instance of a shortest path problem is given as an input graph, $G = (V, E)$, with edge weights $(w_1, w_2, \dots, w_{|E|})$ and a set $S \subseteq V^2$ of pairs of vertices. The objective is to find a subset $P \subseteq E$ of edges that connect all the vertex pairs, $(u, v) \in S$ while minimizing the total weight $\sum_{e \in P} w_e$. The Single Source Shortest Paths Problem (SSSP) and the All-Pairs Shortest Paths Problem (APSP) are the most studied versions of the problem in the context of EAs. In SSSP, the set of vertex pairs to be connected is $S = \bigcup_{v \in V \setminus \{s\}} (s, v)$ for a source vertex $s \in V$ which is given as part of the input. For the all-pairs version, the set of vertex pairs is $S = \{(u, v) \in V^2 \mid u \neq v\}$, which means that all the vertex pairs should be connected.

We chose the shortest path problem not only because it is a classical graph problem, but also because its different variations (all-pairs and single source) have been analyzed in the context of evolutionary algorithms. The first result for the single-source shortest path problem in the literature was presented by Scharnow et al. [22]. The prior analysis was later ramified to cover different evolutionary algorithms and variations of the shortest path problem [2, 11, 10, 8].

Scharnow, Tinnefeld and Wegener proved that the single source shortest path problem (SSSP) is not solvable in polynomial time by an EA, if infeasible solutions are penalized by infinitely large weights [22]. The infeasible solutions constituted a large subset of the search space in which an EA cannot navigate. The authors proposed to divide the objective function into separate functions each evaluating the path to a different destination vertex. For a graph with n vertices, the resulting landscape with $n - 1$ objectives (one for each non-source vertex) provided the algorithm with the required gradient towards better solutions even when some of the vertices are not connected to the source. This so-called "multi-objectivized" function allowed an EA to solve the SSSP in expected $\mathcal{O}(n^3)$ time. Baswana et al. complemented the negative result for the single-objective function later by showing that if the penalty of an unfeasible path is set to nw_{max} , then the $(1 + 1)$ EA solves the single objective version of the problem in expected $\mathcal{O}(n^3 \log(n + w_{max}))$ time [2]. For the multi-objectivized function, Doerr et al. established a tight run time of $\Theta(n^2 \max(\log(n), \ell))$ where ℓ is the largest number of edges in a shortest path that connects the source to any vertex [9]. In order to reflect the effect of graph density

on expected optimization time, Doerr and Johannsen introduced an edge-based solution representation and variation operator in contrast to vertex-based representation which was used in previous works. Consequently, the performance of edge-based EAs in sparse graphs, where the number of edges is $m = o(n^2)$, was proved to be asymptotically better than the vertex-based EA [7].

The first run time analysis of an EA on the APSP problem was presented with the main contribution that this problem is the first non-artificial problem where an algorithm with a crossover operator can optimize asymptotically faster than a simple EA without crossover [10]. At the time of this result the genetic algorithms (EAs with crossover operators) were proved to be essential only on artificial pseudo-Boolean problems [17, 18, 16]. In their work on the APSP problem, the authors proved a $\Omega(n^4)$ lower bound on the optimization time of the EA without crossover while the EA that uses crossover has an optimization time of $\mathcal{O}(n^{3.5}\sqrt{\log n})$. This run time analysis considered an initial phase where the mutation operator finds the shortest paths of smaller number of edges, and a second phase where crossover takes over and produces longer paths by combining the smaller shortest paths provided by the mutation operator. Doerr and Theile contributed the tight run time of expected $\Theta(n^{3.25} \log^{1/4}(n))$ by demonstrating that crossover coarsely populates the set of optimal paths and mutation fills the gaps [8]. Further work on the ASPSP problem introduced changes in the crossover operators, either by repairing the offspring solution, or by selecting parents in order to ensure a feasible solution [11]. With these assisting mechanisms, the runtimes reduced to $\mathcal{O}(n^{3.2} \log^{1/5}(n))$ and $\mathcal{O}(n^3 \log(n))$ for the algorithms with repair mechanism and parent selection respectively.

The transition from the analysis of pseudo-Boolean problems to analysis of classical combinatorial optimization problems was an important milestone in the run time analysis of evolutionary algorithms. The polynomially solvable shortest path problems were one of the initial stepping stones for this transition. In order to handle these more complex combinatorial optimization problem, the field of run time analysis diverted from its conventional approach of using the simplest possible algorithms. This convention was a by-product of keeping the scope of the field on toy problems whose optimization are not a real challenge to the practical applications of evolutionary algorithms. However, when more relevant combinatorial optimization problems are analyzed like shortest path problem, the convention of analyzing simple algorithms was left behind. Whenever the simple algorithms had difficulties in solving a problem, the algorithmic components in problem specific algorithms were incorporated into EAs piece by piece. However certain related problems remain open. Since the previous results were dependent on the monotonic increase in objective function, the performance of non-elitist algorithms with stochastic selection was unknown. While in [10, 8] and [11] a population based algorithm with a crossover operator was used, neither the population nor the crossover operator was conventional in the sense that the population consisted of a single complete solution divided into independent components and the crossover operator recombined these components to create a new offspring. Therefore, the performance of an algorithm with a “population”—in the sense that a set of self-contained solutions and a crossover operator that recombines information from multiple solutions—has never been analyzed in terms of running time.

8.3 Method: Level-Based Analysis

The level-based theorem was proposed as a generalization of the theorem provided by Lehre [19] and later refined by Dang and Lehre [5] for the purpose of analyzing non-elitist population based algorithms. The theorem considers the progress of a search process through a given partitioning of the search space, \mathcal{X} with the optimal solutions set $\mathcal{X}^* \subset \mathcal{X}$, into m level sets $(\mathcal{X}, A_2, \dots, A_{m-1}, \mathcal{X}^*)$ which are nested in the sense that $\mathcal{X} \supset A_2 \supset A_3 \supset \dots \supset A_{m-1} \supset \mathcal{X}^*$. The results in prior works were limited to algorithms with unary variation operators and the solution space partitioning had to be fitness-based i.e. $f(a) \leq f(b)$ for all $a \in (A_i \setminus A_{i+1})$ and $b \in A_{i+1}$ [19, 5]. In the generalized version of the theorem presented by Corus et al. [3], the unary variation operator in the former theorem is replaced with a mapping D , which maps a population of size λ to a probability distribution over the search space \mathcal{X} . For a standard genetic algorithm the mapping D is the combined effect of a sequence of stochastic selection, recombination and variation operators but it can also incorporate the stochasticity of noisy or partially evaluated fitness functions [6, 4]. Nevertheless, the distribution D allows the level-based theorem to be valid for all population based algorithms which independently samples each individual in the population P_{t+1} from the same distribution $D(P_t)$.

Algorithm 1 Population-based algorithm with independent sampling

- 1: Finite state space \mathcal{X} , and population size $\lambda \in \mathbb{N}$,
 - 2: Mapping D from \mathcal{X}^λ to the space of probability distributions over \mathcal{X} .
 - 3: $P_0 \sim \text{Unif}(\mathcal{X}^\lambda)$
 - 4: **for** $t = 0, 1, 2, \dots$ until termination condition met **do**
 - 5: Sample $P_{t+1}(i) \sim D(P_t)$ independently for each $i \in [\lambda]$
 - 6: **end for**
-

The termination condition for the Algorithm 1 is unspecified since the runtime is defined as the time until the optimal solution is sampled for the first time assuming that the algorithm runs forever. A more realistic termination condition can be set by limiting the number of iterations with a value larger than the expected run time. The following theorem considers population-based algorithms at a very abstract level outlined in Algorithm 1, such that all the components like fitness evaluations, variations and selection mechanisms are replaced by a single distribution $D(P_t)$ over the solution space \mathcal{X} . As shown in earlier work leading to the level-based theorem [20], the identical independent sampling of solutions from this distribution $D(P_t)$ allows the use of concentration of measure results from probability theory.

Theorem 8.1 (Theorem 1 in [3]). *Given a partition (A_1, \dots, A_{m+1}) of \mathcal{X} , define $T := \min\{t\lambda \mid |P_t \cap A_{m+1}| > 0\}$ to be the first point in time that elements of A_{m+1} appear in P_t of Algorithm 1. If there exist parameters $z_1, \dots, z_m, z_* \in (0, 1]$, $\delta > 0$, a constant $\gamma_0 \in (0, 1)$ and a function $z_0 : (0, \gamma_0) \rightarrow \mathbb{R}$ such that for all $j \in [m]$, $P \in \mathcal{X}^\lambda$, $y \sim D(P)$ and $\gamma \in (0, \gamma_0)$ we have*

$$(G1) \quad \Pr\left(y \in A_j^+ \mid |P \cap A_{j-1}^+| \geq \gamma_0 \lambda\right) \geq z_j \geq z_*$$

$$(G2) \quad \Pr\left(y \in A_j^+ \mid |P \cap A_{j-1}^+| \geq \gamma_0 \lambda, |P \cap A_j^+| \geq \gamma \lambda\right) \geq z_0(\gamma) \geq (1 + \delta)\gamma$$

$$(G3) \quad \lambda \geq \frac{2}{a} \ln\left(\frac{16m}{ac\varepsilon z_*}\right) \text{ with } a = \frac{\delta^2 \gamma_0}{2(1 + \delta)}, \varepsilon = \min\{\delta/2, 1/2\} \text{ and } c = \varepsilon^4/24$$

$$\text{then } E[T] \leq \frac{2}{c\varepsilon} \left(m\lambda(1 + \ln(1 + c\lambda)) + \sum_{j=1}^m \frac{1}{z_j}\right)$$

The first two conditions require some guarantee of progress and selective pressure from the distribution D . The first condition requires a certain probability z_j of creating an individual at level $j + 1$ when some fixed fraction γ_0 of the population is already at level j or higher, and the second condition requires that the high quality solutions will increase in number with a multiplicative factor of $(1 + \delta)$. The final condition is the minimum population size for the theorem to be valid in terms of the lower bounds achieved in previous conditions.

Algorithm 2 Non-elitist genetic algorithm

```

Initialize  $P_0$ 
 $t := 0$ 
while Termination conditions are not met do
  for  $i = 1$  to  $\lambda$  do
     $p_1 \sim \text{selection}(P_t)$ 
     $p_2 \sim \text{selection}(P_t)$ 
     $P_{t+1}(i) := \text{mutation}(\text{crossover}(p_1, p_2))$ 
  end for
   $t := t + 1$ 
end while

```

In order to simplify the use of the theorem, a corollary is provided in [3] that translates the above requirements to the context of genetic algorithms. Since the scope is narrowed, the distribution $D(P_t)$ is divided into selection, crossover and mutation operators but the details of these operators were not specified so that any algorithm in the form of Algorithm 2 can be analyzed with the following corollary.

Corollary 8.1 (Corollary 1 in [3]). *Given a function $f : \mathcal{X} \rightarrow \mathbb{R}$ and a partition (A_1, \dots, A_{m+1}) of \mathcal{X} , let $T := \min\{t\lambda \mid |P_t \cap A_{m+1}| > 0\}$ be the runtime of the non-elitist Genetic Algorithm, as described in Algorithm 2, on f . If there exist parameters $s_1, \dots, s_m, s_*, p_0, \varepsilon_1 \in (0, 1]$, $\delta > 0$, and a constant $\gamma_0 \in (0, 1)$ such that for all $j \in [m]$, $P \in \mathcal{X}^\lambda$, and $\gamma \in (0, \gamma_0)$*

$$(C1) \quad p_{\text{mut}}(y \in A_j^+ \mid x \in A_{j-1}^+) \geq s_j \geq s_*$$

$$(C2) \quad p_{\text{mut}}(y \in A_j^+ \mid x \in A_j^+) \geq p_0$$

$$(C3) \quad p_{\text{xor}}(x \in A_j^+ \mid u \in A_{j-1}^+, v \in A_j^+) \geq \varepsilon_1$$

$$(C4) \quad \beta(\gamma, P) \geq \gamma \sqrt{\frac{1 + \delta}{p_0 \varepsilon_1 \gamma_0}}$$

$$(C5) \quad \lambda \geq \frac{2}{a} \ln \left(\frac{32mp_0}{(\delta\gamma_0)^2 c s_* \psi} \right) \text{ with } a := \frac{\delta^2 \gamma_0}{2(1+\delta)}, \psi := \min\left\{\frac{\delta}{2}, \frac{1}{2}\right\} \text{ and } c := \frac{\psi^4}{24}$$

$$\text{then } E[T] \leq \frac{2}{c\psi} \left(m\lambda(1 + \ln(1 + c\lambda)) + \frac{p_0}{(1+\delta)\gamma_0} \sum_{j=1}^m \frac{1}{s_j} \right).$$

Here, $\beta(\gamma, P)$ is the probability that the selection mechanism chooses an individual from P that is at least as good as the individual with rank $\lceil \gamma\lambda \rceil$ in terms of level. The conditions (C1) and (C4) on the mutation and selection operators satisfy the first condition (G1) in the main theorem as the responsibility of improving solutions is delegated to mutation while the fourth condition on selection mechanism implies that having sufficient number of solutions at level j leads to a high probability of one of them being selected for mutation. A similar breakdown is possible for conditions (C2)–(C4) since they together satisfy the second condition (G2) in the main theorem. The lower bound ε_1 is the probability that the crossover operator produces an offspring solution in a level which is strictly higher than its worst parent. The lower bound p_0 is the probability of not worsening the solution quality. The final condition gives the required population size similarly to the main theorem.

Moreover, in the following Lemma 8.1 from [3], the parameter settings that satisfy condition (C4) for k -tournament-, exponential ranking-, and (μ, λ) -selection are provided.

Lemma 8.1 ([3]). *For any constant $\delta > 0$, there exists a constant $\gamma_0 \in (0, 1)$ such that*

1. *k -tournament selection with $k \geq 4(1 + \delta)/(\varepsilon_1 p_0)$ satisfies (C4)*
2. *(μ, λ) -selection with $\lambda/\mu \geq (1 + \delta)/(\varepsilon_1 p_0)$ satisfies (C4)*
3. *exponential ranking selection with $\eta \geq 4(1 + \delta)/(\varepsilon_1 p_0)$, satisfies (C4).*

8.4 Design of a Genetic Algorithm

The corollary to the level-based theorem for genetic algorithms produces an upper bound on the expected runtime (Algorithm 2) if the algorithm satisfies certain requirements when optimizing a problem. However, when an arbitrary pair of algorithm and problem is analyzed, often some of these conflicting requirements are not satisfied. Here we will show that when the theory is leading the design of the algorithm, it is easy to meet the requirements while keeping some flexibility for the algorithm.

As a general rule, we aim to keep the term $\sum_{i=1}^m 1/s_j$ in the expected optimization time as small as possible when establishing the level structure. The levels should be few in numbers and close enough in terms of improvement probability. Furthermore, two probabilities are significant for the performance of genetic algorithms: the probability p_0 that the output of mutation operator will preserve or improve the level of the parent solution (C2), and the probability ε_1 that crossover operator will

generate a solution at a higher level than its worst parent (C3). Since the corollary holds for any algorithm of the form in Algorithm 2, the degree of freedom during the design of the algorithm includes the choices of mutation and crossover operators, the selection mechanism and even the objective function.

While we propose operators for each role in Algorithm 2, the algorithm will still have some flexibility. Because when the goal is an asymptotic runtime result for the algorithm then Corollary 8.1 only requires asymptotic bounds on the parameters $\{s_j\}_{j \in [m]}$, p_0 , and ε_1 . This allows the algorithm designer to hybridize the necessary operators proposed in this section with any other operator with constant probability (i.e. using the necessary operator with probability p and the other operator with $1 - p$ for any constant $p \in (0, 1)$) without changing asymptotically the upper bound on the expected runtime. Such a hybridization can be used to tackle extraordinary situations in which the main operator has a very small probability of improving the current solution.

In this section we will first design an algorithm for the single-source version of the shortest path problem and later discuss the necessary alteration for tackling the all-pairs version.

8.4.1 Representation of Solutions

We represent a candidate solution for the single-source shortest path problem as $n - 1$ sequences composed of elements of $V \setminus \{s\}$. In the literature mentioned above different representations that use preceding vertices and sets of edges were used. In our representation which is a sequence of vertices, the source and destination vertices are omitted. A path from the source vertex s to a target vertex v_u that visits vertex v_k just after the source and vertex v_r just before the destination is represented as $P^u = (v_k, \dots, v_r)$.

We can initialize the population by setting each path $P^i := \emptyset$ which means the only edge in the path is (s, v_i) . This initial solution is not only easy to build but also helps skipping the time until the algorithm finds the shortest paths with single edges. Since we will still need to consider these levels during analysis the initial population plays no role design-wise. Any other initialization procedure can replace the suggested one as long as it is constituted of feasible solutions.

8.4.2 The Objective Function and Level Structure

The single-source shortest path problem has a natural objective function which simply sums the weights of all the edges in all the paths. However, the level structure used in the theorem do not have to correlate with the objective function values and a more complex objective function that will guide the algorithm can be designed without further complicating the runtime analysis.

The first step is to convert the sequence of vertices to a set of edges. This conversion is straightforward, for the sequence $P^j = (v_{\pi(1)}, v_{\pi(2)}, v_{\pi(3)}, \dots)$ the edge set is $E^j = \{(s, v_{\pi(1)}), (v_{\pi(1)}, v_{\pi(2)}), \dots\}$ and the total weight is $W^j = \sum_{e \in E^j} w_e$.

If we were able to check whether the current total weight W^j of the path to vertex v_j is optimal, we could count the number of optimal W^j 's for $j \in \{0, 1, \dots, n-1\}$ as a coarse way to quantify the quality of solutions. However, we can not tell whether W^j is optimal or not during the run of the algorithm. Instead, whenever we want to compare two solutions x_1 and x_2 , we first sort the vectors of total weights W_1 and W_2 in ascending order. Then we compare the total weights of the first (shortest) paths in each vector in sorted vectors W_1^π and W_2^ϕ . Whichever weight is smaller, the corresponding solution is considered a better solution when the algorithm ranks the individuals in the population. We will refer to the weights of the second shortest paths $W_2^{\phi(2)}$ and $W_1^{\pi(2)}$ only if $W_1^{\pi(1)}$ and $W_2^{\phi(1)}$ are equal and to the third shortest paths only if the tie can not be broken by the second shortest paths either. The index i of comparison ($W_1^{\pi(i)}, W_2^{\phi(i)}$) is increased at each tie until $W_1^{\pi(i)} \neq W_2^{\phi(i)}$ i.e. the tie is broken. The solutions are considered to be of equal quality if all the comparisons result in a tie. This objective function, although complicated, requires only $O(n \log n)$ basic operations to sort the vector and it will allow the genetic algorithm to simulate the well known Djisktra's algorithm for the shortest path problem. Djisktra's algorithm fixes the smallest shortest path in each iteration and with the help of this objective function we will force the population to converge on the shortest paths in the same order. Since the sub-paths of optimal paths are also optimal, we will use the converged optimal paths to build other paths that requires more edges.

At this point we design a partition of the search space into levels. In order to achieve a good bound we need levels large enough to keep the number of levels small and small enough to ensure that even the worst solutions in any level has a certain probability of being improved to the next one. When shaping the levels of the search space we will make use of the optimality of sub-paths as we did in the objective function. However, during analysis we can use the information that is not normally available to the algorithm, like the optimal paths in a candidate solution. W.l.o.g. consider a unique optimal solution $P_* = \{P_*^1, P_*^2, \dots, P_*^{n-1}\}$ and $P_*^\pi = (P_*^{\pi(1)}, P_*^{\pi(2)}, \dots, P_*^{\pi(n-1)})$ where P_* is sorted such that $W_*^{\pi(i)} \leq W_*^{\pi(i+1)} \forall i \in [n-1]$. So, we divide the solution space \mathcal{X} into n levels, where for all $j \in \{0\} \cup [n-1]$, the level A_j consists of all the solutions that contains $\{P_*^{\pi(i)} \mid 1 \leq i \leq j\}$, the smallest j shortest paths of the optimal solution. This partitioning of the search space ignores the total length of the edges in the candidate solution and focuses only on the paths that are optimal. Moreover, adding an optimal path contributes to the level of the solution only if all of the shorter optimal paths are already set correctly.

8.4.3 The Mutation Operator

With a fixed level structure, the choice of mutation operator will determine the upgrade probabilities s_j , $j \in [n]$ required by condition (C1) and p_0 required by condition (C2), the probability that the mutation operator increases or maintains the quality of the input solution.

Operator 3 Mutation(x)

Input: A solution x consisting of k sequences of vertices x^1, x^2, \dots, x^k
 Pick $s \sim \text{Pois}(1)$
for $i = 1$ to s **do**
 For $m \sim \text{Unif}([k])$
 if with probability $1/k$ **then**
 Set $x^m := \emptyset$
 else
 Set $x^m := (x^j, v_j)$ for $j \sim \text{Unif}([k])$
 end if
end for
Return: x

Consider the mutation operator presented in Operator 3, which as a local operation selects a path x^j , uniformly at random, to be modified either by removing all the vertices in the path with probability $1/n$ or with probability $1 - 1/n$ replaced by another randomly selected path x^k with the destination vertex v_k appended at the end. First of all, since the number of local operations S is distributed according to $\text{Pois}(1)$, the probability that S is equal to zero is $1/e$. Since $S = 0$ implies that the solution is not changed at all, we have $p_0 \geq 1/e$, which satisfies the (C2). For condition (C1), we need to prove that this operator can improve any solution from level j to level $j + 1$ at least with some probability s_j . Note that if a solution is in level j , the smallest j paths in the optimal solution are already set correctly. For improving to level $(j + 1)$ it is sufficient to set the $(j + 1)$ th smallest optimal path, $P_*^{\pi(j+1)}$, correctly, without tempering with any of the first correctly set j paths. If this path has more than one edge, i.e. it visits at least one more vertex v_r before the destination of $P_*^{\pi(j+1)}$, consider the path P_*^r to vertex v_r in the optimal solution. The path P_*^r is shorter than the path $P_*^{\pi(j+1)}$ because the edge weights are positive. Then if our solution is on level j , it has the optimal sub-path up to any vertex v_r that needs to be visited before the destination of $P_*^{\pi(j+1)}$. The mutation operator picks $S = 1$ with probability $1/e$, then with probability $1/(n - 1)$, $x^{\pi(j+1)}$ in the current solution is picked for modification. If $x^{\pi(j+1)}$ is replaced instead of being erased and the correct subpath of $P_*^{\pi(j+1)}$ is chosen to replace the path $x^{\pi(j+1)}$, then in the offspring solution $\text{mut}(x)^{\pi(j+1)} = P_*^{\pi(j+1)}$ with probability $1/(n - 1)^2(1 - 1/n) = \Omega(1/n^2)$. Since $\text{mut}(x)^{\pi(j+1)} = P_*^{\pi(j+1)} \wedge S = 1$ implies that $\text{mut}(x) \in A_{j+1}$, we can bound $s_j \geq \frac{1}{e(n-1)^2}(1 - \frac{1}{n}) = \Omega(\frac{1}{n^2})$ When $P_*^{\pi(j+1)}$ has a single edge $(s, v_{\pi(j+1)})$ the mu-

tation operator has to remove all the vertices in $x^{\pi(j+1)}$. If $x^{\pi(j+1)}$ is selected for modification with probability $1/(n-1)$ then with probability $1/n$ the mutation operator removes all the vertices in the path. In total the probability that $mut(x) \in A_{j+1}$ is similarly $s_j \geq \frac{1}{en(n-1)} = \Omega(\frac{1}{n^2})$

8.4.4 The Crossover Operator

It is shown in some of the earlier works on EAs and the all-pairs shortest path problem that the crossover operator helps creating paths that involve more edges and allows a multiplicative growth in the number of optimal paths discovered. However, in the context of Corollary 8.1, the contribution from the crossover operator is to ensure that individuals with average quality in the solution will be improved if they are recombined with one of the high quality solutions. The lower bound ε_1 in condition (C3) of Corollary 8.1 is the guiding value when deciding how our crossover will work. Consider two solutions $x^1 \in A_j$ and $x^2 \in A_{j+1}$, i.e., x^1 and x^2 share the j shortest paths in the optimal solution and x^2 alone has the $(j+1)$ -th shortest path. Our condition implies that the offspring solution should have all these j paths and the $(j+1)$ -th path, so that it will be on a level strictly higher than its worse parent, x^1 . Therefore, the crossover operator should keep all the shared j paths and pick the $(j+1)$ -th path from parent x^2 with a reasonable probability. This cannot be accomplished by the crossover operators that concatenates the paths as in [12] since even though they allow large improvements they also have a large probability of worsening the solution. A simpler crossover on the other hand, which picks each path from either one of the parents with probability $1/2$ will result in a constant ε_1 . This is because the shared paths will be copied to the offspring and the extra path that makes the difference between the parents will be selected from the better parent with probability $1/2$.

Operator 4 Crossover(x^1, x^2)

Input: Two solutions x and y each consisting of k sequences of vertices

for $i = 1$ to k **do**

if with probability $1/2$ **then**

 Set $z^i := x^i$

else

 Set $z^i := y^i$

end if

end for

Return: z

8.4.5 Other Parameter Settings

In the context of Algorithm 2, fixing the mutation and the crossover operators mostly concludes the design of the algorithm. The remaining parameters like the selection mechanism and the population size can be directly set according to the result of the Corollary 8.1 and Lemma 8.1.

The selection mechanism for our algorithm can be chosen among some of the widely used non-elitist selection mechanisms, such as k -tournament selection, exponential ranking selection, and (μ, λ) -selection. The required parameters for each of these selection mechanisms are given in Lemma 8.1. Since the parameters p_0 and ε_1 are both constant, we can conclude that the required parameters k , η and μ/λ are also constant.

By condition (C5), the population size must satisfy $\lambda \geq \frac{2}{a} \ln \left(\frac{32mp_0}{(\delta\gamma_0)^2 cs_* \psi} \right)$. This means that the minimum population size depends logarithmically on the inverse of smallest improvement probability, the number of levels m , p_0 and ε_1 . Since the latter two are constant and the smallest improvement probability is in $\Omega(1/n^2)$ we conclude that a population size of $\lambda \in \mathcal{O}(\log n)$ satisfies condition (C5).

8.4.6 All-Pairs Shortest Path Problem

In this section we will adapt the genetic algorithm we designed for SSSP to the all-pairs version of the shortest path problem. The obvious change is in the problem representation necessary for SSSP's algorithm to be applied to the APSP. The complete solution of APSP problem consists of $n(n-1)$ sequence of vertices between $n(n-1)$ pairs of vertices. The source and the destination vertices are omitted from the sequence and the sequence of vertices are converted to a set of edges in a similar fashion as in SSSP. For the sequence $P^{ij} = (v_{\pi(1)}, \dots, v_{\pi(\ell)})$ of length ℓ , the edge set is $E^{ij} = \{(v_i, v_{\pi(1)}), (v_{\pi(1)}, v_{\pi(2)}), \dots, (v_{\pi(\ell-1)}, v_{\pi(\ell)}), (v_{\pi(\ell)}, v_j)\}$ and the total weight is $W^{ij} = \sum_{e \in E^{ij}} w_e$. Similar changes in level structure of the search space necessary for the purpose of analysis. We divide the solution space \mathcal{X} into $m = n(n-1) + 1$ levels, where for all $j \in \{0\} \cup [n(n-1) + 1]$, the level A_j consists of all the solutions that contains the smallest j shortest paths of the optimal solution.

The solution comparison procedure, mutation operator, and crossover operators are identical to the ones used for SSSP. The identical operators maintain the same values for parameters p_0 and ε_1 . For APSP, the lower bound on the improvement probabilities s_i are $\Omega(1/n^3)$ since the correct mutation step involves picking the right path among $n(n-1) + 1$ alternatives (rather than n) and picking the correct vertex to append among $\mathcal{O}(n)$ alternatives.

8.5 Expected Running Time

As promised above, we provide the runtime of our algorithm on the single-source shortest path problem.

Theorem 8.2. *The non-elitist GA in Algorithm 2 using the crossover operator given in Operator 4, and the mutation operator given in Operator 3, and either k -tournament selection with $k \geq 8e(1 + \delta)$, or (μ, λ) -selection with $\lambda/\mu \geq 2e(1 + \delta)$ or exponential ranking selection with $\eta \geq 8e(1 + \delta)$, for a constant $\delta > 0$, and population size $\lambda \geq c \ln n$ for some constant $c > 0$, has expected runtime $O(n^3)$ on the single-source shortest path problem.*

Proof. We show that the algorithm satisfies the conditions of Corollary 8.1. Condition (C1) is satisfied for $s_* \in \Omega(1/n^2)$. Condition (C2) is satisfied for $p_0 \geq 1/e$ by the Operator 3. Condition (C3) is satisfied for $\varepsilon_1 \geq 1/2$ by the Operator 4. Condition (C4) is satisfied for some constant γ_0 by the given parameters in Theorem 8.2 due to Lemma 8.1. Finally, condition (C5) is satisfied since $m = n$, $s_* \in \Omega(1/n^2)$, $p_o \in \mathcal{O}(1)$ and $\varepsilon_1 \in \mathcal{O}(1)$. The expected runtime $\mathcal{O}(n^3)$ is obtained by plugging in the parameters m , s_j , p_0 and ε_1 to the expression in the Corollary 8.1.

The same result also holds for the all-pairs version with the modification to improvement probabilities and the number of levels.

Theorem 8.3. *The non-elitist GA in Algorithm 2 using the crossover operator given in Operator 4, and the mutation operator given in Operator 3, and either k -tournament selection with $k \geq 8e(1 + \delta)$, or (μ, λ) -selection with $\lambda/\mu \geq 2e(1 + \delta)$ or exponential ranking selection with $\eta \geq 8e(1 + \delta)$, for a constant $\delta > 0$, and population size $\lambda \geq c \ln n$ for some constant $c > 0$, has expected runtime $O(n^5)$ on the all-pairs shortest path problem.*

Proof. We show that the algorithm satisfies the conditions of Corollary 8.1. Condition (C1) is satisfied for $s_* \in \Omega(1/n^3)$. Condition (C2) is satisfied for $p_0 \geq 1/e$ by the Operator 3. Condition (C3) is satisfied for $\varepsilon_1 \geq 1/2$ by the Operator 4. Condition (C4) is satisfied for some constant γ_0 by the given parameters in Theorem 8.3 due to Lemma 8.1. Finally, condition (C5) is satisfied since $m = n(n - 1) + 1$, $s_* \in \Omega(1/n^3)$, $p_o \in \mathcal{O}(1)$ and $\varepsilon_1 \in \mathcal{O}(1)$. The expected runtime $\mathcal{O}(n^5)$ is obtained by plugging in the parameters m , s_j , p_0 and ε_1 to the expression in the Corollary 8.1.

We may want to compare our result with the performance of EA without crossover since the elementary crossover operator we use is fundamentally different than the crossover operators used. The upper bound of $\mathcal{O}(n^3)$ is close to the performance of elitist EA. Still, we note that it does not reflect the exact optimization time in terms of elementary operations. There is a missing $\Theta(\ln n)$ factor when we only count the number of function evaluations since the comparison mechanism relies on sorting the paths in the solution. Since this sorting procedure is not used in the elitist algorithms that solves the problem, we can conclude that our upper bound is worse by a logarithmic factor.

Even with the alteration to the algorithm, the discrepancy between the upper bounds on expected runtime of the elitist EA and non-elitist GA's is larger for APSP than it is for SSSP. The expected runtime is $\Theta(n^4)$ for EA's without crossover on APSP [10]. If we were to compare this result with the tight runtime of $\Theta(n^3)$ for SSSP, we see a multiplicative factor of n which is admissible since the all-pairs shortest path problem can be divided into n SSSP problems with n different source vertices. However, we will observe a looser upper bound of $\mathcal{O}(n^5)$ for APSP while the upper bound we provided in the previous chapter is $\mathcal{O}(n^3)$.

In the EA proposed for solving APSP, only a single path is altered between two objective function evaluations. This allows an objective function to distinguish between improvements and deterioration even when the objective function aggregates the weights of all paths. However, when two arbitrary solutions are compared the aggregate objective function value can be deceptive. This loss of precision is critical for the non-elitist algorithm since unless higher level solutions are favored by the objective function, the progress over levels cannot be sustained.

The underlying idea of the $\mathcal{O}(n^4)$ upper bound for the elitist EA is that the mutation operator can improve any path at most n times before that path becomes optimal. A corresponding level structure for the non-elitist case would be the total number of improvements required for reaching the optimal solution. With this level structure it is not guaranteed that a solution at a better level will always be favored by the objective function. A similar obstacle was present for the level structure used in the previous section for SSSP. However, for the proposed level structure of SSSP, being on level j required having "all" the j optimal paths with smallest total weights, not having a total of j optimal paths. The comparison procedure which sorts the paths in order of increasing total weight and comparing the shortest paths first guaranteed that solutions of higher level are preferred. For the aggregate number of necessary improvements how such a method might work is not apparent. So we restrict ourselves to the same level structure used in previous section for the analysis of APSP as well, even though the number of levels is liable for the increase in the upper bound on expected optimization time.

8.6 Conclusion

In this paper we reversed conventional runtime analysis and designed an algorithm which fits the analytical tool, in this case level-based analysis [3]. By doing so, we obtained a rigorously proven performance result by simply plugging in the parameters of the resulting algorithm. We hope that this work will set an example for how practitioners can apply theoretical methods to design provably efficient algorithms without going through tedious analysis. Moreover, we showed that using a high level runtime tool as a design guide can ease the experimental burden, not only for evaluating performance but also for setting parameters since the theorem provides the required population size and specific parameters for selection mechanism.

We saw that the essential design decisions were the addition of a subroutine that allows bulk deletion of vertices in a path and the choice of a simple and stable crossover. We exploited the strength of our analysis tool and incorporated the problem specific information into our objective function that allowed us to simulate a well-known deterministic algorithm without making the runtime analysis any harder. We also kept some degrees of freedom for the algorithm by allowing the use of hybridized operator and a variety of selection mechanisms.

On the theoretical side, this paper considers a non-elitist population based algorithm, which was hard to analyze before the introduction of the level-based theorem and previously not considered for many popular problems, including the single-source and all-pairs shortest path problems. Moreover, we provided the first results that uses a binary crossover operator on the shortest path problem, which is an important result since the all-pairs version of this problem is known to be the first non-artificial problem that asymptotically benefited from the use of crossover operator. The next step in this research is to extend this procedural construction to other combinatorial optimization problems. On the higher level, the corollary for genetic algorithms could be improved to take the contribution of the crossover into consideration when lower bounding the improvement probability.

Acknowledgements This research received funding from the European Union Seventh Framework Programme (FP7/2007–2013) under grant agreement no 618091 (SAGE) and from the EP-SRC under grant agreement no EP/M004252/1.

References

1. A. Auger, B. Doerr, *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, vol. 1 (World Scientific, Singapore, 2011)
2. S. Baswana, S. Biswas, B. Doerr, T. Friedrich, P.P. Kurur, F. Neumann, Computing single source shortest paths using single-objective fitness functions, in *Foundations of Genetic Algorithms (FOGA '09)* (ACM Press, New York, 2009), pp. 59–66
3. D. Corus, D.-C. Dang, A.V. Eremeev, P.K. Lehre, Level-based analysis of genetic algorithms and other search processes, in *Proceedings of the 13th International Conference on Parallel Problem Solving from Nature (PPSN 2014)* (Springer International Publishing, Ljubljana/Slovenia, 2014), pp. 912–921
4. D.-C. Dang, P.K. Lehre, Evolution under partial information, in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2014)*, pp. 1359–1366 (2014)
5. D.-C. Dang, P.K. Lehre, Refined upper bounds on the expected runtime of non-elitist populations from fitness-levels, in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2014)* (2014), pp. 1367–1374
6. D.-C. Dang, P.K. Lehre, Efficient optimisation of noisy fitness functions with population-based evolutionary algorithms, in *Foundations of Genetic Algorithms (FOGA '15)* (ACM, New York, 2015), pp. 62–68
7. B. Doerr, D. Johannsen, Edge-based representation beats vertex-based representation in shortest path problems, in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2010)* (ACM, New York, 2010), pp. 759–766

8. B. Doerr, M. Theile, Improved analysis methods for crossover-based algorithms, in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2009)* (ACM, New York, 2009), pp. 247–254
9. B. Doerr, E. Happ, C. Klein, A tight bound for the (1+1)-EA on the single source shortest path problem, in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007)* (2007), pp. 1890–1895
10. B. Doerr, E. Happ, C. Klein, Crossover can provably be useful in evolutionary computation, in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2008)* (ACM, New York, 2008), pp. 539–546
11. B. Doerr, D. Johannsen, T. Kötzing, More effective crossover operators for the all-pairs shortest path problem, in *Parallel Problem Solving from Nature, (PPSN 2010)* (Springer, Berlin, 2010), pp. 184–193
12. B. Doerr, E. Happ, C. Klein, Crossover can provably be useful in evolutionary computation. *Theor. Comput. Sci.* **425**, 17–33 (2012)
13. S. Droste, T. Jansen, K. Tinnefeld, I. Wegener, A new framework for the valuation of algorithms for black-box optimization, in *Foundations of Genetic Algorithms (FOGA '02)* (Morgan Kaufmann, San Francisco, CA, 2003), pp. 253–270
14. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Number 2 (Addison-Wesley, Reading, MA, 1989)
15. T. Jansen, *Analyzing Evolutionary Algorithms: The Computer Science Perspective* (Springer Science and Business Media, Berlin, 2013)
16. T. Jansen, I. Wegener, On the analysis of evolutionary algorithms – a proof that crossover really can help, in *Proceedings of 7th Annual European Symposium on Algorithms (ESA 99)*. Lecture Notes in Computer Science, vol. 1643 (Springer, New York, 1999), p. 700
17. T. Jansen, I. Wegener, On the analysis of evolutionary algorithms – a proof that crossover really can help. *Algorithmica* **34**(1), 47–66 (2002)
18. T. Jansen, I. Wegener, Real royal road functions – where crossover provably is essential. *Discret. Appl. Math.* **149**(1–3), 111–125 (2005)
19. P.K. Lehre, Fitness-levels for non-elitist populations, in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2011)* (2011), pp. 2075–2082
20. P.K. Lehre, X. Yao, On the impact of mutation-selection balance on the runtime of evolutionary algorithms. *IEEE Trans. Evol. Comput.* **16**(2), 225–241 (2012)
21. F. Neumann, C. Witt, *Bioinspired Computation in Combinatorial Optimization* (Springer, Heidelberg, 2010)
22. J. Scharnow, K. Tinnefeld, I. Wegener, The analysis of evolutionary algorithms on sorting and shortest paths problems. *J. Math. Model. Algorithm.* **3**(4), 349–366 (2005)

Chapter 9

On the Impact of Representation and Algorithm Selection for Optimisation in Process Design: Motivating a Meta-Heuristic Framework

Eric S. Fraga, Abdellah Salhi, and El-Ghazali Talbi

Abstract In an ideal world, it would be straightforward to identify the most suitable optimisation method to use in the solution of a given optimisation problem. However, although some methods may be more widely applicable than others, it is impossible a priori to know which method will work best. This may be due to the particular mathematical properties of the mathematical model, i.e. the formulation. It may also be due to the representation of the variables in the model. This combination of choices of method, representation and formulation makes it difficult to predict which combination may be best.

This paper presents an example from process engineering, the design of heat exchanger networks, for which two different representations for the same formulation are available. Two different heuristic optimisation procedures are considered. The results demonstrate that any given combination will not lead to the best outcome across a range of case studies. This motivates the need for a multi-algorithm, multi-representation approach to optimisation, at least for process design.

Keywords Representation • Optimisation • Genetic algorithm • Simulated annealing • Heat exchanger networks

E.S. Fraga (✉)

Department of Chemical Engineering, Centre for Process Systems Engineering, UCL, Torrington Place, London WC1E 7JE, UK

e-mail: e.fraga@ucl.ac.uk

A. Salhi

University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK

e-mail: as@essex.ac.uk

E.-G. Talbi

INRIA Laboratory, CRISTAL/CNRS, University Lille 1, Villeneuve d'Ascq, France

e-mail: el-ghazali.talbi@univ-lille1.fr

9.1 Introduction

It is well known that some algorithms perform better than others on a given problem. The performance of algorithms may even be affected by the instance of the problem solved. For instance, an unstable algorithm which otherwise is suitable for some problem, may perform poorly on an instance that involves ill-conditioned data. What is less known is that different mathematical problem *formulations* of the same problem affect differently the solution process, too. Hall and McKinnon [10] present some examples for the Simplex method. It is, therefore, important to choose not only the most suitable algorithm for a given problem, but one that is most suitable for a given formulation of the problem and the particular instance being solved. The issue of different instances can be dealt with via tailoring, as illustrated recently for the flexible flow-shop problem [20]. In the presence of alternate formulations of the same problem and a variety of possible algorithms, the match making problem is as hard as the original optimization problem. In fact, it is potentially much harder as it reduces to the *Halting Problem* and hence is not *computable*. Nevertheless, in practice, it is a problem that must be addressed and the solution of it is beneficial.

9.2 Representation

As well as algorithms and formulations, there is also the issue of *representation*: the mapping of a data structure, in a form suitable for encoding on a computer, to a *state* in the given mathematical formulation. The representation may often be defined in the context of a particular algorithm yet the encoding is generally independent of the algorithm. Given an optimisation or search problem, with a specified mathematical formulation, a representation is a finite description of an element in the solution space of the problem and a data structure to hold it. An element could be complex: e.g. it might consist of sub-elements in a hierarchical structure. For instance, in genetic algorithm, a particular optimisation variable x may be represented using a real-valued allele or by a binary representation which discretises the domain of x more coarsely; in a genetic program, the encoding could be a tree, represented using linked lists, which maps to a mathematical expression.

A representation defines a search space implicitly. This space is independent of the solution algorithm used. The representation may therefore have a significant impact on the efficacy of the search and the efficiency of the algorithm for the specific problem as formulated. Often, the difference in the quality of solution obtained and in the computational performance between two representations for a genetic algorithm, for instance, will depend on the closeness of match between the space defined by the representation and the behaviour of any objective function in the space defined by the mathematical formulation. In optimisation based computer aided process design, an example is the use of string encodings to represent mixtures in a process instead of a vector of real numbers, leading to significant improvements in search performance [3].

Furthermore, for the particular combination of an algorithm and a representation, there may be associated a number of operations used by the algorithm that manipulate representations, e.g. mutation and crossover operators in a genetic algorithm and the definition of a neighborhood for simulation annealing. A given representation will support, or at the very least encourage, certain operators but not others. These operators will affect how a search space is traversed and whether, in fact, all possible solutions can be reached from any starting point.

9.3 Motivating Example

The premise is that given a problem formulation, whether known in closed form or not, and one or more algorithms, the representation of potential solutions will affect the performance of each algorithm, potentially in different and not necessarily predictable ways. This section presents a motivating example of alternative representations to illustrate the impact of representations on a search algorithm.

Consider the optimisation problem

$$\max_x f(x) = - \left((x - 0.9)^2 \right)^{0.1} \tag{9.1}$$

with $x \in [0, 1]$. The objective function, $f(x)$, shown in Fig. 9.1, is non-smooth and has a maximum value to the right of the centre of the domain. The figure is somewhat misrepresentative as the maximum value is 0 at $x = 0.9$ but the gradient is such that a plotting algorithm has difficulty resolving points near the maximum.

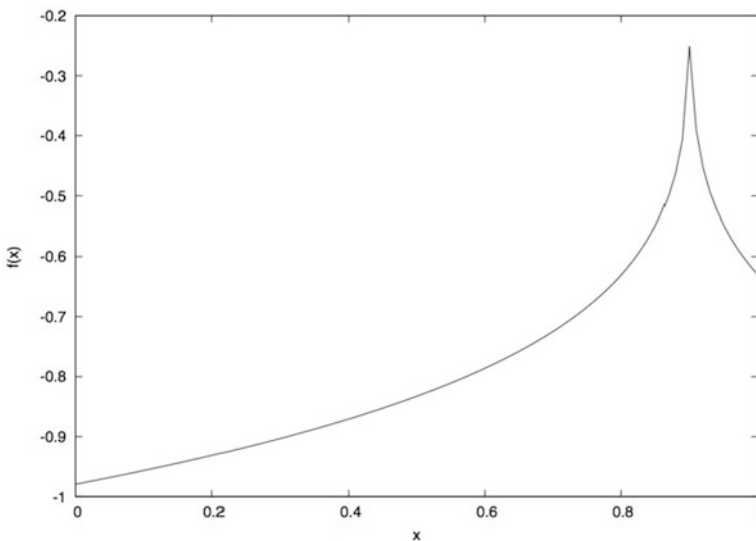


Fig. 9.1 Objective function, Eq. (9.1) for illustrating importance of representation

Although it may be possible to *formulate* this problem differently, it is the representation of the formulation's decision variable, x , that we will concentrate on. In the representations below, we will use $a \in [0, 1]$ to be the decision variable. This variable must be mapped to the decision variable in the formulation, $x \in [0, 1]$, for the evaluation of the objective function.

We define two mappings $m_1 : a \mapsto x$, specifically $x \leftarrow a$ and $m_2 : a \mapsto a^{0.2}$. Both map $a \in [0, 1] \rightarrow x \in [0, 1]$. One mapping, m_1 , is linear and the other is nonlinear. The latter has a shape shown in the following Fig. 9.2.

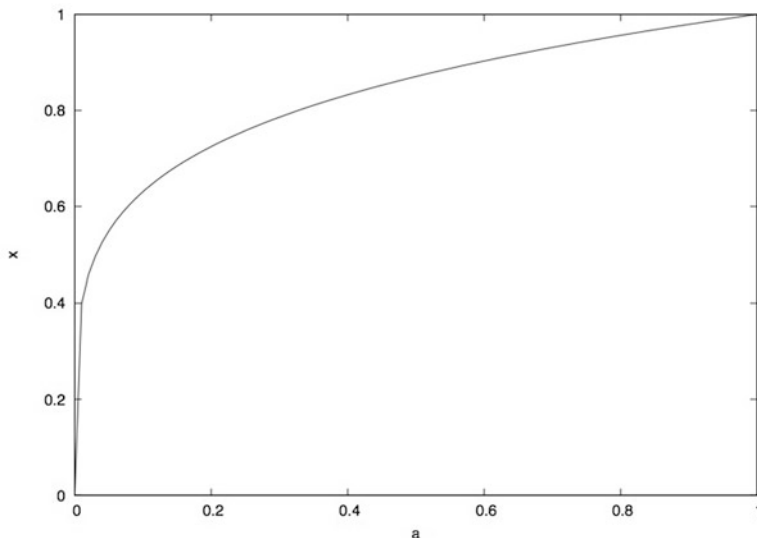


Fig. 9.2 Nonlinear mapping of decision variables from representation a to formulation x : $m_2 : a \mapsto a^{0.2}$

If a random search algorithm is applied to the search domain $a \in [0, 1]$, we get the results shown in Table 9.1. The results are from 1000 attempts at solving the problem with each representation. For each attempt, the random search procedure generates 100 random points, based on a uniform random number generator, for $a \in [0, 1]$. The table shows that the nonlinear representation, in this case, finds better solutions on average.

Table 9.1 Comparison of linear and nonlinear representations for simple objective function

Representation	Best	Average	Standard deviation
Linear	-0.072622	-0.316067	0.075118
Nonlinear	-0.067555	-0.250395	0.057662

Although this example may appear contrived, in chemical process modelling there are many situations where different representations such as illustrated here

may be appropriate. For instance, key physical properties, such as vapour pressure, have *log* based behaviour resulting from the integration of an inverse function. Knowledge of this behaviour has been used previously in the representation of the relationship between vapour pressure and temperature in the discretisation based dynamic programming method for optimal process design [5].

9.4 The Match-Making Problem

The problem of matching algorithms (*aka* methods) to formulations has been described previously [19]. The combination of a single algorithm with a single formulation (SASF) was extended to the cases of a single algorithm with multiple formulations (SAMF) and for multiple algorithms with multiple formulations (MAMF). An argument was made for the development of procedures which could address the MAMF case so as to identify the best combination of algorithm and formulation for a given problem. This paper argues for the further need to incorporate representation as well as formulation in this matching process.

With this in mind, and since a heuristic can be seen as an algorithm that does not guarantee optimality in finite time, it is easy to see how equivalent solution paradigms to the above can be introduced: MAMR, for instance, would be the most general and powerful paradigm which will take all available algorithms and all available representations and solve both the original problem and the match making problem since, here also, a given representation may not be ideal for a given algorithm. This assumes a single fixed problem formulation.

For the scope of this paper, we have limited the optimisation algorithms to simple implementations of two traditional stochastic methods: simulation annealing (SA) and genetic algorithms (GA). These have both been implemented in the Jacaranda system [7]. We use these to show the need for a MAMR paradigm for process design through a problem of industrial interest.

9.5 Heat Exchanger Network Design

In large chemical processes, from food processing through to bulk chemical production and refining, there is often the potential to use excess heat in one part of the process to meet heating demands in another part. The task of designing the network of heat exchangers which can transfer heat from one part of the process to another is a challenging optimisation task.

A *heat exchanger network synthesis* (HENS) design problem is defined by set of hot streams, those which have excess heat, and a set of cold streams, those which need heating. Each stream is defined by an inlet temperature, a target temperature, and a heat flux which specifies the rate at which the heat must be removed or obtained from the inlet temperature to the target temperature. The result of the design

is a network which consists of exchangers, each of which transfer heat from a hot stream to a cold stream. Each stream may have more than one exchange. The order of exchanges affects the network as does the size of each exchanger.

The core of the heat exchanger network model is the estimation of the heat exchange area for each exchanger based on the relationship between the heat flux, q , the overall heat transfer coefficient, U , the area, A and the temperature driving force, ΔT_{LMTD} :

$$q = UA\Delta T_{\text{LMTD}} \quad (9.2)$$

ΔT_{LMTD} is the *log-mean temperature difference* and represents the effective driving force across the whole exchanger based on the inlet and target temperatures of both streams, hot and cold. This driving force temperature is often defined by

$$\Delta T_{\text{LMTD}} = \frac{\Delta T_2 - \Delta T_1}{\ln \Delta T_2 - \ln \Delta T_1} \quad (9.3)$$

The objective function for design is typically an annualised cost which includes the capital cost of the exchangers and the cost of utilities to meet any leftover demands after heat integration. The capital cost is typically a function of area:

$$C_c = \alpha + \beta A^\gamma \quad (9.4)$$

with $\gamma \in [0, 1]$ but usually on the order of $\gamma = 0.65$ and with $\alpha > 0$. The utility costs are a function of the heat fluxes and the temperatures and there is often a set of utilities at different temperatures available.

The end result is a mixed-integer nonlinear programme (MINLP) which is computationally challenging: the search space is combinatorial in nature and the objective function is non-convex. Although attempts have been made to solve such problems using deterministic solution algorithms [9, 15, 11], scalability and convexity issues have led to the use of stochastic or meta-heuristic methods for solving these problems [2, 4, 13, 14, 16, 17, 18]. There are many forms of meta-heuristic methods proposed for heat exchanger network synthesis; they all share the need to define a problem specific representation for the efficient and effective search of the solution space.

9.6 Representations and Algorithms for HENS

We have developed two representations for heat exchanger network synthesis problems:

SGA is a simple chromosome encoding, i.e. representation, for HENS intended for use by a genetic algorithm suitable for embedding within a large optimisation problem and solution procedure [6]. The aim was simplicity of the encoding. A solution is represented by a fixed number, n , of possible exchanges between hot and cold streams. The representation is a vector of integer values, $y_i, i = 1, \dots, n$,

where $y_i \in [0, n_c \times n_h]$, n_c is the number of cold streams and n_h is the number of hot streams. A value $y_i = c \times h$ indicates the specific match between cold stream $c \in [1, n_c]$ and hot stream $h \in [1, n_h]$. A value of $y_i = 0$ indicates no match.

$$\boxed{y_1} \boxed{y_2} \cdots \boxed{y_n}$$

The algorithm for evaluating a given solution is *greedy*: each possible exchange is evaluated in turn along the chromosome, from left to right, and, if the exchange is feasible, an exchanger is designed to transfer as much heat as possible from hot stream to cold stream in the match identified.

dfHEN is a discrete fixed representation for heat exchanger networks. The simple representation above uses a greedy algorithm to evaluate a design. This may lead to sub-optimal results. A different representation, derived from the method proposed by Lewin [12], covers the space of possible designs more fully. The representation includes the possible matches, grouped in levels, and the amount of exchange to undertake for any given match. There is some redundancy of information in the encoding: a match amount may be 0 which is equivalent to that match not being considered. Each level consists of n_c integer values indicating the hot stream from which to get heat, with a 0 value indicating no exchange at this level. For each match, there is also an integer value which represents the amount of heat to exchange: $q = y_i \delta q$. The potential is there for identifying better solutions but the space represented is larger and has redundancy so there can be a loss of efficiency in the search.

The two representations are generic and independent of the optimisation method, or algorithm, which uses them. To investigate the MAMR situation, we have considered two stochastic optimisation procedures: GA, a simple genetic algorithm [6], and SA, a simulated annealing approach [8]. The details of these methods are not critical as the point is to investigate whether the combinations of algorithm and representation can lead to different results for different problems. Tuning parameters for these methods is also not relevant as such tuning typically depends on the specific optimisation problem, further emphasising the need for matching between methods and representations.

9.7 Results

A broad selection of heat exchanger network design problems has been assembled from the literature: CS1 is the 4SP problem from [16], CS2 is example 6 from [1], CS3-CS5 are problems A through C from [12], CS6 is the example from [15] and CS7 the example from [21]. All of these problems were solved using both representations (SGA and dfHEN) and with two simple methods, GA and SA, implemented in the Jacaranda system [7], leading to four different combinations.

Table 9.2 summarises the results obtained. The four combinations for each case study are allowed the same amount of computational resource. For each row, the resource allocated, in terms of objective function evaluations, is that amount sufficient

to allow at least one of the combinations to achieve the best known solution for the design problem.

The first observation is that the simulated annealing procedure performs as well as or better than the genetic algorithm in almost all cases. The second observation, and arguably the more important one, is that no single representation is best overall. The SGA representation performs best for case studies 1–3 and 7 but the dfHEN representation wins out for case studies 4–6. There is nothing immediately obvious in the problem formulations that would allow one to predict which representation would be most appropriate. It is also worth noting that GA + dfHEN combination has the least worst relative performance (1.57); all other combinations have at least one case which performs worse relatively.

Table 9.2 Summary of performance of two optimisation methods with two different representations for the seven case studies

Case study	GA dfHEN	GA SGA	SA dfHEN	SA SGA
1	1.57	1.03	1.75	1.00
2	1.01	1.00	1.02	1.00
3	1.09	1.01	1.02	1.00
4	1.52	2.47	1.02	2.23
5	1.55	2.48	1.01	2.30
6	1.06	1.20	1.01	1.20
7	1.16	1.16	1.16	1.00

Each entry is the average value of the best objective function value obtained over 10 runs relative to the value of the best solution obtained overall. The best average result for each case study is emboldened

9.8 Conclusions

The possibility of multiple formulations for an optimisation problem, the choice of representations for the degrees of freedom and the range of solvers available leads to the difficult task of choosing the correct combination. This paper has demonstrated that the choice does matter. For a set of problems, all in the same domain of heat exchanger network synthesis, different combinations of method and representation work best for individual problems. This motivates the development of an over-arching method which could identify the best combination and solve the problem most effectively. We propose a Multiple Heuristics, Multiple Representation (MHMR) paradigm which mirrors the Multiple Algorithm, Multiple Formulation (MAMF) model for the exact solution [19]. Exploring this paradigm, say through the design and implementation of prototype software frameworks will be the focus for future work in our respective research groups.

References

1. C.S. Adjiman, I.P. Androulakis, C.A. Floudas, Global optimization of mixed-integer nonlinear problems. *AIChE J.* **46**, 1769–1798 (2000)
2. G. Athier, P. Floquet, L. Pibouleau, S. Domenech, Process optimization by simulated annealing and NLP procedures. Application to heat exchanger network synthesis. *Comput. Chem. Eng.* **21**, S475–S580 (1997)
3. E.S. Fraga, Discrete optimization using string encodings for the synthesis of complete chemical processes, in *State of the Art in Global Optimization: Computational Methods & Applications*, ed. by C.A. Floudas, P.M. Pardalos (Kluwer, Dordrecht, 1996), pp. 627–651
4. E.S. Fraga, A rewriting grammar for heat exchanger network structure evolution with stream splitting. *Eng. Optim.* **41**, 813–831 (2009)
5. E.S. Fraga, K.I.M. McKinnon, The use of dynamic programming with parallel computers for process synthesis. *Comput. Chem. Eng.* **18**, 1–13 (1994)
6. E.S. Fraga, A. Žilinskas, Evaluation of hybrid optimization methods for the optimal design of heat integrated distillation sequences. *Adv. Eng. Softw.* **34**, 73–86 (2003)
7. E.S. Fraga, J. Hagemann, A.D. Estrada Villagrana, I.D.L. Bogle, Incorporation of dynamic behaviour in an automated process synthesis system. *Comput. Chem. Eng.* **24**, 189–194 (2000)
8. E.S. Fraga, R. Patel, G.W.A. Rowe, A visual representation of process heat exchange as a basis for user interaction and stochastic optimization. *Chem. Eng. Res. Des.* **79**, 765–776 (2001)
9. K.C. Furman, N.V. Sahinidis, A critical review and annotated bibliography for heat exchanger network synthesis in the 20th century. *Ind. Eng. Chem. Res.* **41**, 2335–2370 (2002)
10. J.A.J. Hall, K.I.M. McKinnon, The simplex examples where the simplex method cycles and conditions where expand fails to prevent cycling. *Math. Program.* **100**, 133–150 (2004). doi:10.1007/s10107-003-0488-1
11. A.J. Isafiade, D.M. Fraser, Interval based MINLP superstructure synthesis of heat exchanger networks for multi-period operations. *Chem. Eng. Res. Des.* **88**, 1329–1341 (2010)
12. D.R. Lewin, A generalized method for HEN synthesis using stochastic optimization – II. The synthesis of cost-optimal networks. *Comput. Chem. Eng.* **22**, 1387–1405 (1998)
13. B. Lin, D.C. Miller, Solving heat exchanger network synthesis problems with Tabu Search. *Comput. Chem. Eng.* **28**, 1451–1464 (2004)
14. X. Luo, Q.Y. Wen, G. Fieg, A hybrid genetic algorithm for synthesis of heat exchanger networks. *Comput. Chem. Eng.* **33**, 1169–1181 (2009)
15. W. Morton, Optimisation of a heat exchanger network superstructure using nonlinear programming. *Proc. Inst. Mech. Eng. Part E* **216**, 89–104 (2002)
16. A. Pariyani, A. Gupta, P. Ghosh, Design of heat exchanger networks using randomized algorithm. *Comput. Chem. Eng.* **30**, 1046–1053 (2006)
17. M.A.S.S. Ravagnani, A.P. Silva, P.A. Arroyo, A.A. Constantino, Heat exchanger network synthesis and optimisation using genetic algorithm. *Appl. Therm. Eng.* **25**, 1003–1017 (2005)
18. G.W.A. Rowe, E.S. Fraga, Co-operating ant swarm model for heat exchanger network design, in *Adaptive Computing in Design and Manufacture VII*, ed. by I.C. Parmee (The Institute for People-Centred Computation, Bristol, 2006), pp. 29–35
19. A. Salhi, The ultimate solution approach to intractable problems, in *Proceedings of the 6th IMT-GT Conference on Mathematics, Statistics and its Applications (ICMSA2010)*, Universiti Tunku Abdul Rahman (2010), pp. 84–93
20. A. Salhi, J.A. Vazquez-Rodriguez, Tailoring hyper-heuristics to specific instances of a scheduling problem using affinity and competence functions. *J. Memetic Comput.* (2014). doi:10.1007/s12293-013-012107
21. M. Serna, A. Jimenez, An area targeting algorithm for the synthesis of heat exchanger networks. *Chem. Eng. Sci.* **59**, 2517–2520 (2004)

Chapter 10

Manufacturing Cell Formation Problem Using Hybrid Cuckoo Search Algorithm

**Bouchra Karoum, Bouazza Elbenani, Noussaima El Khattabi,
and Abdelhakim A. El Imrani**

Abstract Cellular manufacturing, as one of the most important applications of Group Technology, has gained popularity in both academic research and industrial applications. The cell formation problem is considered the first and the foremost issue faced in the designing of cellular manufacturing systems that attempts to minimize the inter-cell movement of the products while maximize the machines utilization. This paper presents an adapted optimization algorithm entitled the cuckoo search algorithm for solving this kind of problems. The proposed method is tested on different benchmark problems; the obtained results are then compared to others available in the literature. The comparison result reveals that on 31 out of 35 problems (88.57%) the results of the introduced method are among the best results.

Keywords Cell formation problem • Lévy flight • Cuckoo search • Metaheuristic • Cellular manufacturing

10.1 Introduction

Cellular manufacturing (CM) is one of the most important applications of the Group Technology that aims to convert a production system into several mutually separable production cells. Where dissimilar machines are aggregated into machine groups (also known as manufacturing cells) and similar parts into part families so that one or more part families can be processed within a single machine group. The main objective is minimizing the intercellular and intracellular movements. Many significant

B. Karoum (✉) • B. Elbenani

Research Computer Science Laboratory (LRI), Faculty of Science, Mohammed V University of Rabat, B.P. 1014 Rabat, Morocco

e-mail: bouchra.karoum@gmail.com; elbenani@fsr.ac.ma

N. El Khattabi • A.A. El Imrani

Conception and Systems Laboratory (LCS), Faculty of Science, Mohammed V University of Rabat, B.P. 1014 Rabat, Morocco

e-mail: khattabi@fsr.ac.ma; elimrani@fsr.ac.ma

benefits have been reported for CM, including reducing setup and throughput times, minimizing the material handling costs, improving the quality and the production control, etc.

The cell formation problem (CFP), which is a non-polynomial hard optimization problem [3], is the first issue in the designing of cellular manufacturing systems. It involves the identification of machine cells and part families with the objective of minimizing the intercellular and intracellular part movements.

In the last decades, many solution methods have been proposed to solve this problem. Chandrasekharan and Rajagopalan [1] developed an algorithm entitled ZODIAC (zero-one data: ideal seed algorithm for clustering) for solving the CFP and used a new concept called relative efficiency as a stopping rule for the iterations. Srinivasan and Narendran [14] proposed an efficient non-hierarchical clustering algorithm, based on initial seeds obtained from the assignment method, named GRAFICS for the rearrangement of parts and machines. Elbenani and Ferland [4] suggested an exact method for solving the manufacturing cell formation problem. A good survey paper which review briefly the different methodologies used to solve the problem until 2008 is presented in [10].

Recently, several authors have adopted the use of the metaheuristic algorithms for the cell formation problem, because of its efficiency in solving combinatorial optimization problems. Goncalves and Resende [6] presented a new algorithm that combines a local search heuristic with a genetic algorithm for obtaining machine cells and part families. James et al. [8] employed a standard grouping genetic algorithm with a local search mechanism to form machine-part cells. Diaz et al. [2] suggested a greedy randomized adaptive search procedure (GRASP) heuristic to obtain lower bounds for the optimal solution of the CFP. Sayadi et al. [11] recommended a new solution based on a discrete firefly algorithm for solving the problem. Solimanpur and Elmi [13] presented a nested application of tabu search approach to solve the problem heuristically. Husseinzadeh Kashan et al. [7] introduced a new solution approach based on the particle swarm optimization (PSO) algorithm for solving the CFP. Ying et al. [16] developed a simulated annealing based metaheuristic with variable neighbourhood to form part-machine cells. Elbenani et al. [5] hybridized a genetic algorithm with a local search procedure that applies sequentially an intensification strategy to improve locally a current solution and a diversification strategy destroying more extensively a current solution to recover a new one. A grouping version of league championship algorithm has been proposed by Seyedhosseini et al. [12] for solving the CFP, and so on.

In this paper, a recently developed cuckoo search (CS) algorithm is adopted for solving the CFP with the aim of maximizing the grouping efficacy. The CS algorithm is combined with a local search method in order to intensify the search towards promising regions. The experimental results show that the proposed algorithm generates good results in reasonable computational time.

The remainder of this article is organized as follows: Sect. 10.2 describes the CFP; Sect. 10.3 gives an overview of the standard cuckoo search algorithm and introduces the improvement carried out on the algorithm to solve the CFP. The results of numerical experiments on a set of benchmark problems are reported in Sect. 10.4, and concluding remarks are given in Sect. 10.5.

10.2 Problem Formulation

The input parameter of the CF problem is a machine-part incidence matrix A where each row represents a machine and each column represents a part. The objective is to determine a rearrangement so that machine utilization within a cell is maximized and the inter-cellular movement is minimized. Figure 10.1 shows a 5×5 machine part incidence matrix to a problem with five machines and five parts. The second matrix indicates a solution of the problem by partition into 3 different cells illustrated in the gray blocks. The 1 outside the diagonal blocks are called exceptional elements, while the 0 inside the diagonal blocks are called voids.

Different measures have appeared in the literature to compare the efficiency of the methods. The most used is the grouping efficacy (Eff) [9] where the closer the grouping efficacy is to 1, the better will be the solution obtained (see Eq. (10.1)).

$$Eff = \frac{(a - a_1^{Out})}{(a + a_0^{In})} \quad (10.1)$$

Where:

a : Total number of entries equal to 1 in the matrix A ;

a_1^{Out} : Number of exceptional elements;

a_0^{In} : Number of voids.

To formulate this problem, a mathematical model similar to the one used in [4] is adopted.

Parameters:

i, j, k : Index of machines, parts and cells, respectively.

M, P, C : Number of machines, parts and cells, respectively.

A : Machine-part incidence matrix $A = [a_{ij}]$.

a : Total number of entries equal to 1 in the matrix A .

$a_{ij} = 1$ if machine i process part j ; 0 otherwise.

$x_{ik} = 1$ if machine i belongs to cell k ; 0 otherwise.

$y_{jk} = 1$ if part j belongs to cell k ; 0 otherwise.

The objective function of the CFP can be presented as follow:

$$\min ComEff(x, y) = \frac{a + \sum_{k=1}^C \sum_{i=1}^M \sum_{j=1}^P (1 - 2a_{ij})x_{ik}y_{jk}}{a + \sum_{k=1}^C \sum_{i=1}^M \sum_{j=1}^P (1 - a_{ij})x_{ik}y_{jk}} \quad (10.2)$$

Parts		1	2	3	4	5
Machines	1	0	1	0	0	1
	2	1	0	1	0	0
	3	1	0	0	1	0
	4	0	1	0	0	0
	5	1	0	0	1	0

Initial incidence matrix

Parts		2	5	1	4	3
Machines	1	1	1	0	0	0
	4	1	0	0	0	0
	3	0	0	1	1	0
	5	0	0	1	1	0
	2	0	0	1	0	1

Matrix solution

Fig. 10.1 Incidence matrix solution

Subject to:

$$\sum_{k=1}^C x_{ik} = 1 \quad i = 1, \dots, M \tag{10.3}$$

$$\sum_{k=1}^C y_{jk} = 1 \quad j = 1, \dots, P \tag{10.4}$$

$$\sum_{i=1}^M x_{ik} \geq 1 \quad k = 1, \dots, C \tag{10.5}$$

$$\sum_{j=1}^P y_{jk} \geq 1 \quad k = 1, \dots, C \tag{10.6}$$

$$x_{ik} = 0 \text{ or } 1 \quad i = 1, \dots, M; k = 1, \dots, C \tag{10.7}$$

$$y_{jk} = 0 \text{ or } 1 \quad j = 1, \dots, P; k = 1, \dots, C \tag{10.8}$$

Constraint (10.3) ensures that each machine is assigned to exactly one cell. Constraint (10.4) guarantees that each part must be assigned to only one cell. Inequalities (10.5) and (10.6) ensure that each cell includes at least one machine and one part. Finally, constraints (10.7) and (10.8) denote that the decision variables are binary.

10.3 Improved Cuckoo Search Algorithm

10.3.1 Basic Cuckoo Search

Cuckoo search algorithm is a new nature-inspired metaheuristic algorithm developed by Yang and Deb in 2009 [15]. It was inspired by the special lifestyle and the aggressive brood parasitic behavior of some species of a bird family called cuckoo.

This algorithm is initially designed for solving multimodal functions and can be summarized around the following three ideal rules [15]:

- Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest.
- The best nests with high quality of eggs (solutions) will carry over to the next generations.
- The number of host nests is fixed, and a host can discover an alien egg with a probability $p_a \in [0, 1]$.

The last assumption can be approximated by a fraction p_a of the n nests are replaced by new ones.

In the standard cuckoo search algorithm, the behavior of cuckoo is associated with Lévy flight to search a new nest. Lévy flights, named by the French mathematician Paul Lévy, represent a model of random walks characterized by their step lengths that obey a power-law distribution.

When generating a new solution $x_l^{(t+1)}$ for a cuckoo l , in iteration $t + 1$, a Lévy flight is performed using Eq. (10.9).

$$x_l^{(t+1)} = x_l^{(t)} + \alpha \oplus \text{lévy}(\beta) \quad (10.9)$$

Where $\alpha > 0$ is the step size parameter that should be chosen according to the scales of the problem. In most cases, $\alpha = 1$. The product \oplus means entry-wise multiplications. The Lévy flight is a random walk where the step length follows a Lévy distribution that has an infinite variance with an infinite mean. It allows the exploration of the search space more efficiently as its step length is much longer in the long run. A simple scheme discussed in detail by Yang [15] can be approximated (\sim) by Eq. (10.10):

$$\text{lévy}(\beta) \sim 0.01 \frac{\mu}{|\nu|^{1/\beta}} (x_b^{(t)} - x_l^{(t)}), \quad 1 < \beta \leq 3 \quad (10.10)$$

Where μ and ν are drawn from normal distribution:

$$\mu \sim N(0, \sigma_\mu^2), \quad \nu \sim N(0, \sigma_\nu^2) \quad (10.11)$$

With

$$\sigma_\mu = \left(\frac{\Gamma(1 + \beta) \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \beta 2^{\frac{(\beta-1)}{2}}} \right)^{\frac{1}{\beta}}, \quad \sigma_\nu = 1 \quad (10.12)$$

Where Γ is the standard Gamma function.

10.3.2 The Proposed Cuckoo Search Algorithm

This section introduces the modification made on the standard CS algorithm in order to solve the cell formation problem which is discrete in nature. The algorithm begins

with an initialization phase, where the number of machines M , the number of parts P , the number of cells C , the population size and the fraction p_a of abandoned nests are defined. Then, an initial population of feasible solutions is created. After initialization, the proposed method iterates until a specific number of loops are met as shown in Algorithm 1.

Algorithm 1 The proposed cuckoo search algorithm

Generate an initial population of n host nests x_i ($i = 1, \dots, n$)
while ($t < MaxGeneration$) or (stop criterion) **do**
 Get a cuckoo l randomly by Lévy flights;
 Evaluate its quality according to its grouping efficacy Eff_l ;
 Apply a local search to the generated solution x_l ;
 Choose a nest r among n randomly;
 if ($Eff_{x_l} > Eff_{x_r}$) **then**
 Replace x_r by the new solution;
 end if
 Abandon a fraction p_a of worse nests and build new ones by Lévy flights
 Rank the solutions and find the current best
end while

10.3.2.1 Solution Representation

The encoding adapted in this paper is similar to the one used by Elbenani et al. [5]. Where the problem variable is represented as a vector with length of $P + M$: $(P_1, \dots, P_P | M_1, \dots, M_M)$. Where:

- P_j is the index of the cell including part j .
- M_i is the index of the cell including machine i .

To understand better, consider the following solution of a problem with 7 parts and 5 machines: $(2, 1, 3, 2, 3, 1, 3 | 2, 1, 2, 3, 1)$. This solution includes three cells: cell 1 contains parts $\{2, 6\}$ and machines $\{2, 5\}$, cell 2 contains parts $\{1, 4\}$ and machines $\{1, 3\}$ and cell 3 contains parts $\{3, 5, 7\}$ and machine $\{4\}$.

10.3.2.2 Population Initialization

The initial solutions are generated randomly. Each machine i and part j are assigned randomly to a cell k . each cell must contain at least one machine and one part. To fix infeasibilities that may arise from an empty cell (cell without parts/machines), a repair process is activated. This process involves removing a part/ machine from the cell, including the most to the empty cell, which induces the smallest decrease of the grouping efficacy.

10.3.2.3 Nests

In the CS algorithm, the number of nests is fixed and equal to the population size. A nest is an individual of the population and an abandoned one entails replacing an individual in the population with a new one. Assuming that a cuckoo lays a single egg in one nest; each egg in a nest will be a solution represented by one individual in the population; and an egg of the cuckoo represents a new possible solution.

10.3.2.4 Lévy Flights

Lévy flights represent a model of random walks characterized by their step lengths which obey a power-law distribution as depicted in Eq. (10.10). In this paper, the difference between the two positions ($x_k^{(t)} - x_l^{(t)}$) represent the necessary movements to change from the actual solution given by the second, subtracting term, to the best obtained solution given by the first, subtracted term.

For each of the necessary movements, a random number between 0 and 1 is generated and if its value is less than the correspondent coefficient $Coef = \frac{\mu}{|v|^{1/\beta}}$, the movement is applied to the solution. A part movement coded as (4, 1) represents changing part 4 to cell 1. Similarly for a machine movement.

To understand better, we take the following example, let:

- $Coef = 0.7$;
- The current solution: $x_l = (3, 2, 1, 2, 3, 1, 3, 1 | 1, 2, 3, 1)$;
- The best solution obtained: $x_b = (2, 3, 1, 1, 3, 1, 3, 2 | 1, 2, 3, 3)$.

The necessary part movements, in this case, are (1,2), (2,3), (4,1), (8,2) while the necessary machine movement is (4,3). Five random values are generated and compared with $Coef : 0.8, 0.3, 0.9, 0.6$ and 0.4 . Therefore, the only part movements that would be applied in the current solution are (2,3) and (8,2), besides the machine movement (4,3). The resulting solution will be: $x_l = (3, 3, 1, 2, 3, 1, 3, 2 | 1, 2, 3, 3)$ If the produced solution is unfeasible, the repair process described above is employed.

10.3.2.5 Local Search

In order to improve the quality of the solutions, the CS algorithm is combined with a search mechanism. The local search approach adopted is based on the one introduced by Goncalves and Resende [6]. Since it is simple, uses the same measure to compare the efficiency of the methods and generates good results.

Based on the initial set of machine groups of the incoming solution, each part is assigned to the cell that maximizes the grouping efficacy calculated by Eq. (10.1). If the modified solution is better than the current solution, the modified solution

replaces the current one. Afterwards, the process will restart by the assignment of the machines. This approach iterates by reassignment of parts, then reassignment of machines until the quality of the new solution does not exceed the quality of the last solution.

10.4 Computational Results

The proposed algorithm is tested on a set of 35 benchmark problems collected from the literature [5]. For each test problem, the number of machines (M), parts (P) and cells (C) are shown in Table 10.1. Besides, the minimum (*Min. Sol.*), the average (*Ave. Sol.*) and the maximum (*Max. Sol.*) solutions obtained, and the Best-Known solution (*Best Known Sol.*). Also, the computational time in seconds (*CPU. Time(s)*) required and the percentage of the gap (*Gap(%)*) are reported in Table 10.1, respectively. Preliminary tests showed that the following values are suitable to find best solutions:

- Population size $n = 50$;
- Portion of abandoned nests $p_a = 0.25$;
- Maximum number of iterations = 200.

For each benchmark problem, 10 independent runs of the algorithm with these parameters are performed. Table 10.1 summarizes the obtained results. The described algorithm is coded using Java, and running on a Personal Computer equipped with Pentium (R) Dual Core CPU T4500 clock at 2.30 GHz and 2 GB of RAM.

As shown in Table 10.1, the proposed method generates very good solutions in a reasonable computational time. Since the proposed method can reach the Best-Known solution of 31 benchmark problems. Among these problems, our method attains the Best-Known solution of 17 test problems throughout the 10 runs of the algorithm (*Min. Sol. = Ave. Sol. = Max. Sol.*), which prove the performance of the projected method. Regarding the remaining problems, the average solutions obtained are very close to the Best-Known solutions.

In order to confirm the effectiveness of the proposed method, the percentage of *gap* is calculated with respect to the Best-Known solution of each test problem, using Eq. (10.13). The results shows that the proposed approach has gaps in 4 problems only with the highest one is in problem $P27$ with 2.22% gap, as demonstrated in Table 10.1.

$$Gap = \frac{(Best\ Known\ Solution - Max.\ Sol.)}{Best\ Known\ Solution} \times 100 \quad (10.13)$$

To exhibit potentials of the proposed method, the obtained results are compared with the results of several algorithms developed in the literature for the CFP. Table 10.2 presents the results reported from these algorithms and the best solutions obtained for each problem.

Table 10.1 The numerical results of the proposed algorithm

No.	<i>M</i>	<i>P</i>	<i>C</i>	<i>Min. Sol.</i>	<i>Ave. Sol.</i>	<i>Max. Sol.</i>	<i>BestKnown Sol.</i>	<i>Gap(%)</i>	<i>Time(s)</i>
P1	5	7	2	82.35	82.35	82.35	82.35	0	0.32
P2	5	7	2	69.57	69.57	69.57	69.57	0	0.33
P3	5	18	2	79.59	79.59	79.59	79.59	0	0.35
P4	6	8	2	76.92	76.92	76.92	76.92	0	0.32
P5	7	11	5	60.87	60.87	60.87	60.87	0	0.57
P6	7	11	4	70.83	70.83	70.83	70.83	0	0.46
P7	8	12	4	69.44	69.44	69.44	69.44	0	0.48
P8	8	20	3	85.25	85.25	85.25	85.25	0	0.52
P9	8	20	2	58.72	58.72	58.72	58.72	0	0.40
P10	10	10	5	75.00	75.00	75.00	75.00	0	0.65
P11	10	15	3	92.00	92.00	92.00	92.00	0	0.51
P12	14	24	7	71.64	72.02	72.06	72.06	0	3.70
P13	14	24	7	69.44	71.51	71.83	71.83	0	4.30
P14	16	24	8	53.26	53.26	53.26	53.26	0	5.79
P15	16	30	6	69.53	67.42	68.92	69.53	0	5.20
P16	16	43	8	56.25	56.80	57.53	57.53	0	14.26
P17	18	24	9	55.67	57.21	57.73	57.73	0	8.45
P18	20	20	5	41.61	42.59	43.45	43.45	0	2.89
P19	20	23	7	48.09	50.25	50.81	50.81	0	6.38
P20	20	35	5	77.91	77.91	77.91	77.91	0	5.04
P21	20	35	5	57.14	57.89	57.98	57.98	0	5.12
P22	24	40	7	100.0	100.0	100.0	100.0	0	14.35
P23	24	40	7	85.11	85.11	85.11	85.11	0	14.19
P24	24	40	7	73.51	73.51	73.51	73.51	0	14.65
P25	24	40	11	52.74	53.13	53.29	53.29	0	38.39
P26	24	40	12	47.95	48.19	48.61	48.95	0.69	44.32
P27	24	40	12	43.84	45.49	46.21	47.26	2.22	43.86
P28	27	27	5	54.82	54.82	54.82	54.82	0	5.23
P29	28	46	10	44.87	46.45	47.06	47.23	0.36	50.68
P30	30	41	14	62.42	62.92	63.31	63.31	0	83.68
P31	30	50	13	58.29	59.31	59.77	60.12	0.58	93.43
P32	30	50	14	49.72	50.46	50.83	50.83	0	85.97
P33	36	90	17	45.66	47.08	47.75	47.75	0	532.59
P34	37	53	3	59.26	59.97	60.64	60.64	0	9.58
P35	40	100	10	84.03	84.03	84.03	84.03	0	248.31

The comparator algorithms are ZODIAC method [1], GRAFICS method [14], evolutionary algorithm (EA) [6], simulated annealing (SA) [16], GRASP heuristic [2], genetic algorithm and large neighbourhood search (GA-LNS) [5].

The approaches: ZODIAC, GRAFICS, EA and GRASP do not allow for singletons (cells having less than two machines or two parts) which may affected the quality of the solutions in comparison with other approaches.

As can be seen, the proposed method achieves the best known solutions in 31 test problems out of the existing 35 test problems being 88.57%. The remaining problems in which the method performs a little bit worse than its comparator are 26,

Table 10.2 Performance of the proposed method compared to other approaches

No.	ZODIAC	GRAFICS	EA	SA	GRASP	GA-LNS	Proposed method	Bestknown Sol.
P1	73.68	73.68	73.68	82.35	73.68	82.35	82.35	82.35
P2	56.22	60.87	62.50	69.57	62.50	69.57	69.57	69.57
P3	–	–	79.59	79.59	79.59	79.59	79.59	79.59
P4	–	–	76.92	76.92	76.92	76.92	76.92	76.92
P5	39.13	53.12	53.13	60.87	53.13	60.87	60.87	60.87
P6	–	–	70.37	70.83	70.37	70.83	70.83	70.83
P7	68.30	68.30	68.29	69.44	69.44	69.44	69.44	69.44
P8	85.24	85.24	85.25	85.25	85.25	85.25	85.25	85.25
P9	58.33	58.33	58.72	58.72	58.72	58.72	58.72	58.72
P10	70.59	70.59	70.59	75.00	70.59	75.00	75.00	75.00
P11	92.00	92.00	92.00	92.00	92.00	92.00	92.00	92.00
P12	64.36	64.36	69.86	72.06	69.86	72.06	72.06	72.06
P13	65.55	65.55	69.33	71.83	69.33	71.83	71.83	71.83
P14	32.09	45.52	52.58	53.26	51.96	53.26	53.26	53.26
P15	67.83	67.83	67.83	68.99	67.83	69.53	69.53	69.53
P16	53.76	54.39	54.86	57.53	56.52	57.53	57.53	57.53
P17	41.84	48.91	54.46	57.73	54.46	57.73	57.73	57.73
P18	21.63	38.26	42.94	43.45	42.96	43.45	43.45	43.45
P19	38.96	49.36	49.65	50.81	49.65	50.81	50.81	50.81
P20	75.14	75.14	76.22	77.91	76.54	77.91	77.91	77.91
P21	–	–	58.07	57.98	58.15	57.98	57.98	57.98
P22	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
P23	85.11	85.11	85.11	85.11	85.11	85.11	85.11	85.11
P24	37.85	73.51	73.51	73.51	73.51	73.51	73.51	73.51
P25	20.42	43.27	51.88	53.29	51.97	53.29	53.29	53.29
P26	18.23	44.51	46.69	48.95	47.37	48.95	48.61	48.95
P27	17.61	41.67	44.75	47.26	44.87	46.58	46.21	47.26
P28	52.14	47.37	54.27	54.82	54.27	54.82	54.82	54.82
P29	33.01	32.86	44.37	47.23	46.06	47.06	47.06	47.23
P30	33.46	55.43	58.11	63.31	59.52	63.12	63.31	63.31
P31	46.06	56.32	59.21	59.77	60.00	60.12	59.77	60.12
P32	21.11	47.96	50.48	50.83	50.51	50.83	50.83	50.83
P33	32.73	39.41	42.12	47.14	45.93	47.75	47.75	47.75
P34	52.21	52.21	56.42	60.64	59.85	60.63	60.64	60.64
P35	83.92	83.92	84.03	84.03	84.03	84.03	84.03	84.03

The bold values indicate the solutions equal to the best-known solutions.

27, 29 and 31. For problem 21, data reported in EA and GRASP are inconsistent with the original data in the literature.

The numerical results indicate that the proposed method generates good results for the 35 problems, and that the performance of this method is rather constant on all types of problems ranging from small to large.

10.5 Conclusion

This paper presents in detail how a recently developed heuristic entitled cuckoo search algorithm is operating for solving the manufacturing cell formation problem. A remarkable contribution of the paper is the novelty of applying CS algorithm to this type of combinatorial optimization problems which require adapting its elements appropriately. The proposed method is combined with a local search mechanism in order to intensify the search and improve the quality of the solutions.

The results obtained in the computational experiments carried out show that on 31 out of 35 problems (88.57%) the results of the introduced method are among the best results. These results justify the superior performance of the method compared to other methods collected from the literature.

For future research, the extension of this approach could be applied to other variants of the cell formation problem having additional factors such as the real time manufacturing data, number of routings for each part, etc. Moreover, the proposed problem can be used in various similar clustering problems that involve the assignment of distinct objects in indistinguishable group.

References

1. M.P. Chandrasekharan, R. Rajagopalan, ZODIAC-an algorithm for concurrent formation of part-families and machine-cells. *Int. J. Prod. Res.* **25**, 835–850 (1987)
2. J.A. Diaz, D. Luna, R. Luna, A GRASP heuristic for the manufacturing cell. *Top.* **20**, 679–706 (2012)
3. C. Dimopoulos, A.M.S. Zalzalá, Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons. *IEEE Trans. Evol. Comput.* **4**, 93–113 (2000)
4. B. Elbenani, J.A. Ferland, *CIRRELT: Cell Formation Problem Solved Exactly with the Dinkelbach Algorithm* (CIRRELT, Montreal, 2012), pp. 1–14
5. B. Elbenani, J.A. Ferland, J. Bellemare, Genetic algorithm and large neighbourhood search to solve the cell formation problem. *Expert Syst. Appl.* **39**, 2408–2414 (2012)
6. J.F. Goncalves, M.G.C. Resende, An evolutionary algorithm for manufacturing cell formation. *Comput. Ind. Eng.* **47**, 247–273 (2004)
7. A. Husseinzadeh Kashan, B. Karimi, A. Noktehdan, A novel discrete particle swarm optimization algorithm for the manufacturing cell formation problem. *Int. J. Adv. Manuf. Tech.* **73**, 1543–1556 (2014)
8. T.L. James, E.C. Brown, K.B. Keeling, A hybrid grouping genetic algorithm for the cell formation problem. *Comput. Oper. Res.* **34**, 2059–2079 (2007)
9. C.S. Kumar, M.P. Chandrasekharan, Grouping efficacy: a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. *Int. J. Protein Res.* **28**, 233–243 (1990)
10. G. Papaioannou, J.M. Wilson, The evolution of cell formation problem methodologies based on recent studies (1997–2008): review and directions for future research. *Eur. J. Oper. Res.* **206**, 509–521 (2010)
11. M.K. Sayadi, A. Hafezalkotob, S.G.J. Naini, Firefly-inspired algorithm for discrete optimization problems: an application to manufacturing cell formation. *J. Manuf. Syst.* **32**, 78–84 (2013)

12. S.M. Seyedhosseini, H. Badkoobei, A. Noktehdan, Machine-part cell formation problem using a group based league championship algorithm. *J. Promot. Manag.* **21**, 55–63 (2015)
13. M. Solimanpur, A. Elmi, A tabu search approach for cell scheduling problem with makespan criterion. *Int. J. Prod. Econ.* **141**, 639–645 (2013)
14. G. Srinivasan, T.T. Narendran, GRAFICS- A nonhierarchical clustering algorithm for group technology. *Int. J. Prod. Res.* **29**, 463–478 (1991)
15. X.S. Yang, S. Deb, Cuckoo search via Lévy flights, in *IEEE NaBIC 2009: Proceedings of the World Congress on Nature and Biologically Inspired Computing*, Coimbatore (2009), pp. 210–214
16. K.C. Ying, S.W. Lin, C.C. Lu, Cell formation using a simulated annealing algorithm with variable neighbourhood. *Eur. J. Ind. Eng.* **5**, 22–42 (2011)

Chapter 11

Hybridization of Branch and Bound Algorithm with Metaheuristics for Designing Reliable Wireless Multimedia Sensor Network

Omer Ozkan, Murat Ermis, and Ilker Bekmezci

Abstract Reliability is a key topic for Wireless Multimedia Sensor Networks (WMSNs) design which involves connectivity and coverage issues with node placement. The main contribution of this chapter is to deploy sensor nodes to maximize the WMSN reliability under a given budget constraint by considering terrain and device specifications. The reliable WMSN design with deployment, connectivity and coverage has NP-hard complexity, therefore a new hybridization of an exact algorithm with metaheuristics is proposed. A Branch&Bound (B&B) approach is embedded into Hybrid Simulated Annealing (HSA) and Hybrid Genetic Algorithm (HGA) to orient the cameras exactly. Since the complexity of the network reliability problem is NP-complete, a Monte Carlo (MC) simulation is used to estimate the network reliability. Experimental study is done on synthetically generated terrains with different scenarios. The results show that HGA outperforms the other approaches especially in large-sized sets.

Keywords Wireless multimedia sensor network • Network reliability • Reliable network design • Hybrid metaheuristics • Branch and bound • Simulated annealing • Genetic algorithm

11.1 Introduction

The sensors in a WMSN have the ability to retrieve, process, relay and store video and audio streams, still images, and scalar sensor data. There are different civil and military applications about WMSNs over the last few years in the literature such as; multimedia surveillance sensor networks, storage of potentially relevant activities,

O. Ozkan (✉) • M. Ermis

Turkish Air Force Academy, Industrial Engineering Department, Istanbul, Turkey
e-mail: o.ozkan@hho.edu.tr; m.ermis@hho.edu.tr

I. Bekmezci

Turkish Air Force Academy, Computer Engineering Department, Istanbul, Turkey
e-mail: i.bekmezci@hho.edu.tr

traffic avoidance, enforcement and control systems, advanced health care delivery, automated assistance for the elderly and family monitors, environmental monitoring, person locator services and industrial process control [1]. Providing security on country borders, operating early warning systems, monitoring and observation systems and determining the routes according to a given mission are vitally important particularly for military operations. In this chapter, a new sensor node deployment plan that includes the hybridization of an exact algorithm with metaheuristics is proposed for WMSN reliability maximization.

Deployment of sensor and relay nodes can severely affect the performance of network lifetime, reliability or cost. Multimedia sensor deployment problem includes locating a set of sensor or relay nodes, and setting their parameters (such as heading angle, tilt etc.). The problem is closely related to the Art Gallery Problem, which purposes to guard an art gallery with the minimum number of guards who together can observe the whole gallery. Although 2-D Art Gallery Problem can be solved optimally, it is proved that it is NP-hard in 3-D space [7].

The approximate methods and Genetic Algorithms (GAs) are applicable for restricted versions of this problem [2, 5]. A binary integer program that solves the best camera deployment given in a determined number of cameras is proposed by Zhao et al. [11]. A cost efficient wireless camera sensor deployment strategy for environment monitoring is also studied [10]. Topcuoglu et al. suggested a new GA to locate and utilize the multimedia network for maximizing coverage and minimizing cost [10]. The first efforts on video sensor location problem were only related with the coverage, however after the understanding of WMSN structure, the connectivity and the other vital network related issues are also studied. One of the works about connected coverage for WMSN is proposed by Han [6].

The reliability of sensor networks combines the reliability of connectivity related data distribution and coverage related data acquisition. Along with the connectivity issues, coverage also must be integrated in Wireless Sensor Network (WSN) or WMSN reliability models. Even though, in many works, connected coverage is introduced as Quality of Service (QoS) for WSN [6], it is also considered as a WSN reliability issue. A reliability model that integrates the traditional connectivity based network reliability with the coverage is proposed by Shrestha et al. [9]. Shrestha et al. investigate the WSN reliability as a hierarchical structure [8], which does not cover all WSNs. In [4], the coverage problem for WSN is studied from the reliability point of view so that a series of sensor placement strategies are proposed for maximum reliability and fault tolerance.

Countries that have their own reconnaissance satellite network can easily detect crime or terrorist activities. The intelligence satellites are used to produce very detailed radar images or photographs of small, strategically important locations. However, due to lack of relevant technology, the other countries must either pay a fee or enter into an exclusive information-sharing agreement (or both) if they want to get access to satellite networks. As an alternative and practical solution, those governments fighting crime or terrorism uses reconnaissance aircrafts, UAVs and WMSN to detect illegal activities or terrorist camps. But, technology advances are now enabling far more accurate and reliable imagery using tiny, low battery powered and

inexpensive sensors that can communicate with each other and monitor the terrain to gather information. In this chapter, we propose a new WMSN reliability metric, that is total information gathered, a model for deploying and configuring a set of given sensors on a synthetically generated 2-D terrain for the maximization of overall network reliability and a novel hybrid solution methodology for surveillance of a terrorist camp or region.

In real surveillance applications, 3-D structure of the terrain must be considered for optimal sensor deployment problems. Although our work is based on 2-D terrains, it can be easily extended for 3-D terrains. But, commercial solvers are not efficient in solving realistic size 3-D terrain problem instances due to the reliability calculations for each sensor. In order to find solutions to realistic size problem instances in a reasonable amount of time, we devise hybrid heuristics on 2-D terrains.

The main problem in this chapter is to maximize WMSN reliability under a limited budget. Instead of dealing with the connectivity and coverage based reliability separately, an information gathering concept is introduced by considering connectivity and coverage issues with a deployment plan. In order to produce the solution of the problem, a new hybridization approach, which integrates an exact algorithm with metaheuristics, is proposed. A B&B approach embedded into SA and GA are employed to find the exact orientations of the cameras (i.e. HSA and HGA). Because of the NP-complete structure of network reliability problem, a Monte Carlo simulation is implemented. The proposed method also considers the importance map of the 2-D terrain, node deployment safety, artificial and natural occlusions in order to model a more realistic reliability approach for surveillance missions.

The rest of the chapter is organized as follows. The next section describes the problem formulation including terrain, target, sensor node and relay node features, information gathering reliability calculation and the mathematical model. In Sect. 11.3, the implementation details of the HSA and HGA approaches for the problem are presented. Section 11.4 summarizes the results of the experimental study with respect to terrain, node and target specifications. Finally, Sect. 11.5 concludes this chapter.

11.2 The Problem Definition

The designed WMSN consists of base station (b) placed near by the border points of the terrain (Ter). The b is linked to the command center via a satellite. There are predefined target points (T) on the Ter and the WMSN operates to get reliable data from the T . The Ter is displayed as a 2-D grid based surface, since it is sufficiently realistic and adequately simple for computational tractability. Although there are several different applications that require mobile targeting, there are also some scenarios that can be realized with fixed targeting. In real terrorist camps, the terrorists can be classified as mobile targets. However, there can be some other fixed targets like logistics buildings, communication centers, ammunition store, armory or artilleries. In this paper, we aim to surveillance only fixed targets.

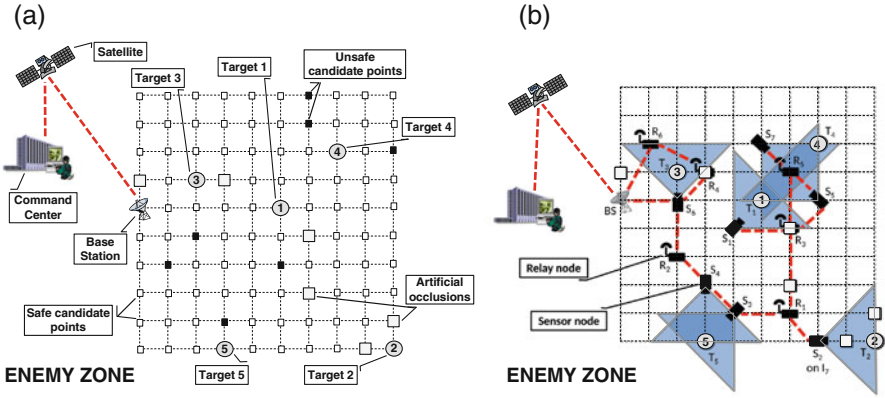


Fig. 11.1 The *Ter* and WMSN properties. (a) A sample *Ter*. (b) A sample WMSN

Table 11.1 The parameters of the problem definition

Component	Notation	Value	Definition
<i>Ter</i>	i	$\in \mathcal{I}$	Candidate point to deploy devices
<i>Ter</i>	x_i	Positive integer	x coordinate of point i
<i>Ter</i>	y_i	Positive integer	y coordinate of point i
<i>Ter</i>	sw_i	[0-1]	Safety weight of point i
<i>Ter</i>	now_i	[0-1]	Natural occlusion weight of point i
<i>Ter</i>	aow_i	[0-1]	Artificial occlusion weight of point i
T	k	$\in \mathcal{T}$	Target
T	iw_k	[0-1]	Importance weight of target k
S	j	$\in \mathcal{S}$	Sensor
S	p_j	Positive integer	Deployment point of sensor j
S	d_j	Real	Depth of view of sensor j
S	v_j	[0-359°]	Viewing angle of sensor j
S	h_j	[0-359°]	Heading angle of sensor j
S	c_j	Real	Cost of sensor j
S	r_j	Real	Communication range of sensor j
S	rel_j	[0-1]	Hardware reliability of sensor j
R	l	$\in \mathcal{R}$	Relay node
R	c_l	Real	Cost of relay l
R	r_l	Real	Communication range of relay l
R	rel_l	[0-1]	Hardware reliability of relay l
-	TB	Real	Total budget

The properties of the *Ter* and T are depicted in Fig. 11.1. The WMSN has sensor (S) and relay (R) nodes connected to the b to obtain the data. The S collect video data from the covered T on the *Ter* and deliver the data to the b via other connected S and R . The R do not have any camera attribution and so their only function is to deliver the received data to the neighbour nodes. A designed WMSN example is presented in Fig. 11.1. The properties of the *Ter*, T , S and R are summarized in Table 11.1.

The iw_k represents the importance weight of a k and if it is 1, it means the k has top influence for the surveillance. Therefore, it has more chance to be covered. If

the iw_k is close to 0, then the k has few importance. The overall of the iw_k on the Ter is the total surveillance data that can be gathered.

Some of the polygons on Ter may have different safety levels. The opponent patrols or soldiers on duty may recognize the sensor or relay node devices easily on some i . In some occasions, the devices can not be placed in to some i because of the geographical structure of the i . The sw_i indicates the safety level of i , and it can be defined as the probability of operating a node safely, when the node is placed on a certain point i .

The occlusion weights (aow_i and now_i) symbolize the occlusion types on the point i . There can be two types of objects located on synthetically generated Ter , which are artificial and natural objects. When an object is located on a polygon, the density value of the object is added to the polygon. These occlusions effects the coverage values of the sensors on targets. The aow_i refers to artificial structures such as buildings, walls, etc. When aow_i is 0, there is no artificial occlusion on i . When aow_i is close to 1, the transparency level of i decreases. Similarly, the now_i involves the natural flora on Ter such as bushes, woodland and forest. The flora decreases the coverage in a rate between upper and lower bounds special to plant density. The decreasing rates are constants and they affect the all points on Ter . The effects of the natural occlusions on the coverage values and the view cone of a sensor j deployed on i are presented in Fig. 11.2.

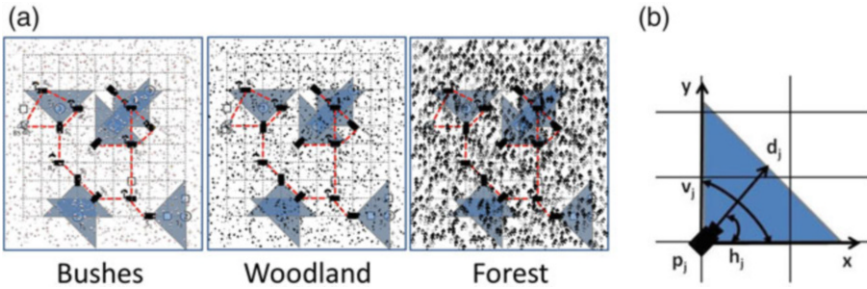


Fig. 11.2 The coverage properties. (a) The effects of natural occlusions. (b) View cone of a sensor

The coverage on a target k by a sensor j is formulated in Eq. (11.1). The $maxocc_{ik}$ is the maximum of the now_i or the aow_i on the line between the node i and target k . $Cone_{ijk}$ is 1, if i is in the view cone of sensor j ; otherwise, it is 0. The node connectivity is based on the Euclidean distances between the nodes. If the distance is less than a certain communication range (r_j or r_i) the nodes are supposed to be connected. Although the obstacles or 3-D structure of the terrain may affect the communication of the nodes, for the simplicity of the calculations, two nodes are assumed as connected if the distance between than is less than a certain communication threshold.

$$Cov_{ijk} = (1 - maxocc_{ik}) * Cone_{ijk} \tag{11.1}$$

Moreover, two reliability definitions are made for the reliability functions of the sensor and relay nodes in WMSNs, respectively. The first reliability definition includes the operation probability of a sensor j on i that collects data from the target k defined in Eq. (11.2).

$$Rel_{ijk} = rel_j * sw_i * Cov_{ijk} \quad (11.2)$$

The second definition presented in Eq. (11.3) refers the operation probability of a relay node l on i . Equations (11.2) and (11.3) are used for the information gathering reliability estimation of the designed WMSN. Equation (11.2) is the information gathering reliability and (11.3) is the transmitting reliability of the nodes.

$$Rel_{il} = rel_l * sw_i \quad (11.3)$$

Ultimately, the WMSN reliability is calculated as the information gathered from the Ter via the deployed S and R over the total information on the Ter . The WMSN reliability is formulated as below in Eq. (11.4). The reliability is represented as $RelIG(S, R, Ter)$ and it involves the gathered information from every covered target k by S and delivered via R and S to the b on Ter over the total information ($\sum_{k \in T} iw_k$) on Ter .

$$\max RelIG(S, R, Ter) = \frac{\sum_{k \in T} \widehat{Rel}(s', r', b, k) * iw_k}{\sum_{k \in T} iw_k} \quad (11.4)$$

s.t.

$$\sum_{i \in I} \left(c_j * \sum_{j \in S} \sum_{m \in O} x_{ijm} + c_l * \sum_{l \in R} x_{il} \right) \leq TB \quad (11.5)$$

$$\sum_{j \in S} \sum_{m \in O} x_{ijm} + \sum_{l \in R} x_{il} \leq 1, \forall i \in I \quad (11.6)$$

$$x_{ijm}, x_{il} \in \{0, 1\}; \forall i \in I; \forall j \in S; \forall m \in O; \forall l \in R \quad (11.7)$$

For each target k covered by s' , the r' and s' are connecting s' to the b on the Ter . The s' is a subset of S and the r' is a subset of R . There are network reliability calculations [$\sum_{k \in T} \widehat{Rel}(s', r', b, k)$ in Eq. (11.4)] of each covered target k by s' . At this point, the exact calculation of network reliability is NP-complete [3]. Hence, a MC simulation method is employed to estimate (\widehat{Rel}). The working nodes in WMSN are determined by random numbers according to Rel_{ijk} and Rel_{il} values in MC simulation. The \widehat{Rel} estimation is done by repeating the simulation 3000 times and calculating the gathered information from the covered targets as the failed nodes are removed from network.

The mathematical model has a budget (TB) constraint given in Eq. (11.5). The other two equations (11.6) and (11.7) are network related design constraints. Equation (11.6) prevents deployment on more than one device on the same polygon i . The x_{ijm} is 1, if camera j has orientation m on polygon i , else it is 0. Also, the x_{il} is 1, if relay node l is on polygon i , else it is 0 in Eq. (11.7).

11.3 Hybridization of Branch and Bound Algorithm with Metaheuristics

The sensor and relay node deployment, node connectivity, coverage of targets and reliable WMSN design has NP-hard complexity, therefore four metaheuristics are proposed. The algorithms are SA, GA, HSA and HGA. The flow charts of the algorithms are depicted in Fig. 11.3.

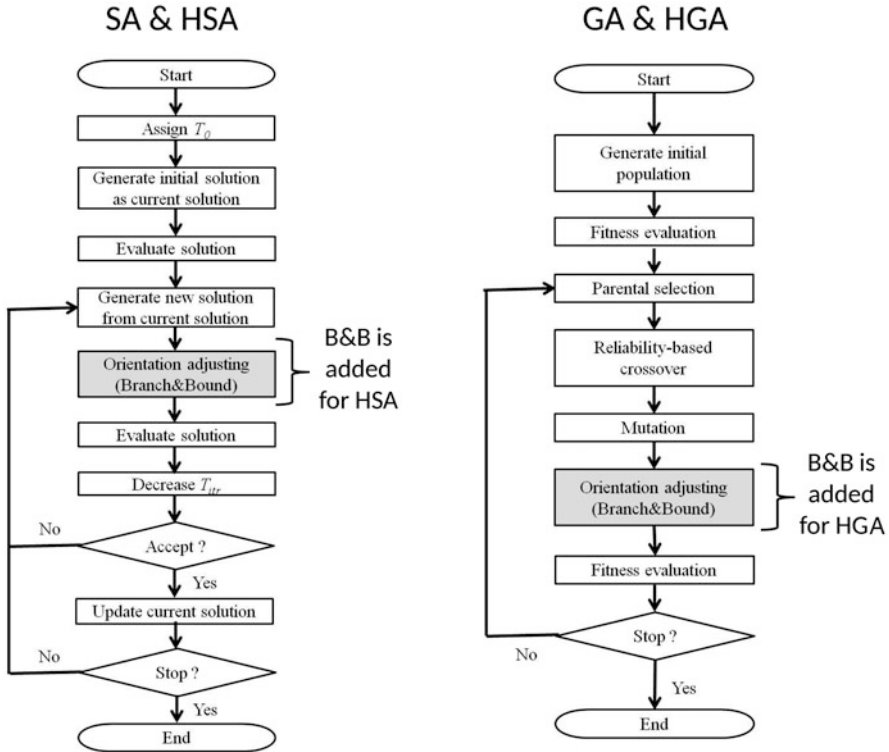


Fig. 11.3 Flowcharts of the proposed algorithms

As figured out from the flow charts, the SA and GA algorithms have the same functions and operators with the hybrid versions except B&B algorithm. The B&B is serial coded and embedded into the hybrid algorithms. Hybridization of metaheuristics with an exact method is preferred to derive benefit from the exact method. Because, the exact methods have capability to advance the solutions for this problem. Thereby, the strong sides of the algorithms are incorporated to eliminate the weak sides of them. The B&B helps the SA and GA to find better solutions for the predefined WMSN problem. The proposed algorithms are described in next subsections below in detail.

11.3.1 Hybridization of Branch and Bound with Simulated Annealing

The SA is an algorithm inspired from the annealing in metallurgy. The SA is chosen because it has effective single solution based searching strategy based on a temperature cooling schedule. The cooling schedule of SA allows for finding better solutions and escaping from local optimum points of the problems.

The proposed SA&HSA has 2-D array type representation and has three division provided in Fig. 11.4. The first division is a row that indicates the randomly selected T and the second division is also a row that covers the I deployed S to get surveillance from T . The last division has multiple rows presenting the connection paths of the selected S to the b via other instruments. TB is a constraint that puts a limit to the sizes of the representation. The representation of the sample WMSN in Fig. 11.1 is also depicted in Fig. 11.4. In the representation sample, the S_5 and S_7 sensors are observing two targets simultaneously.

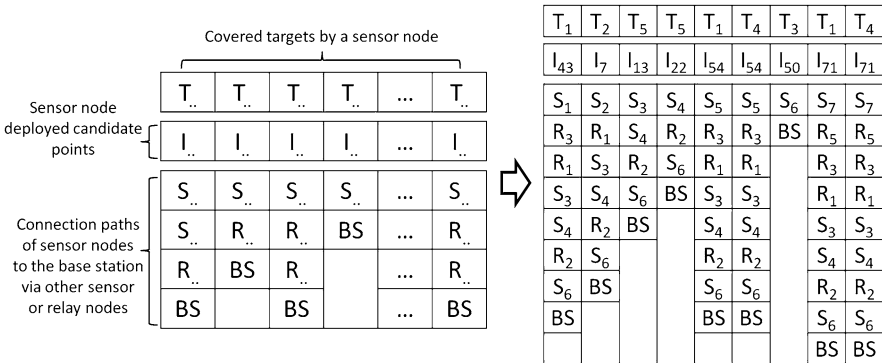


Fig. 11.4 Representation type and the representation of the sample WMSN in Fig. 11.1

The objective function (f) of the SA&HSA is the information gathering reliability estimations of the designed candidate WMSN topologies. The initial solutions of SA&HSA are generated by a heuristic. First of all, a target k and a candidate point i covering that k is selected randomly. The selection chances of k and i are influenced by their selection possibilities. If any k has a higher iw_k , then it has more chance to be selected. Similarly, if an i has a higher Rel_{ijk} value, it has much more probability to be chosen than the other points. Then, a sensor j is deployed to the selected point i . A shortest path heuristic is used to find the placement points of the R nodes to connect that j to the b . Adding S and R nodes process proceeds until satisfying TB constraint.

The initial temperature (T_0) is a key parameter for SA based algorithms. Therefore, the T_0 value for the proposed algorithms is assessed by a Markov chain. The cooling schedule parameters are also important for algorithm performance. The decrease in the temperature parameter is archived by " $T_{t+1} = \alpha * T_t, (t=0,1,2,...)$ " formulation. The stopping condition of the SA&HSA is the maximum iteration number ($maxitr$). The algorithms stop when the iterations reach to the $maxitr$ number.

The *maxitr* number is divided into β number of divisions and in every division the cooling formulation is executed. T_0 , α , β , and *maxitr* parameters are tuned in experimental study.

The neighbourhood function of the algorithms is simply deleting-adding (replacing) a randomly selected sensor j and its connections to the b procedure. The process continues until spending all TB .

After the traditional SA functions, a combinatorial B&B method is inserted serially into HSA to find the exact orientations of the sensors. It is assumed that in the definition of the algorithms, one sensor is watching just one target. The h_j is fixed to an angle that the sensors are observing the selected one target directly. Even so, a camera can cover more than one target simultaneously if the targets are in its coverage zone as depicted in Fig. 11.5. The B&B method is providing new orientations for the sensors by adapting their h_j and aiming to increase the information gathering reliability of the candidate network.

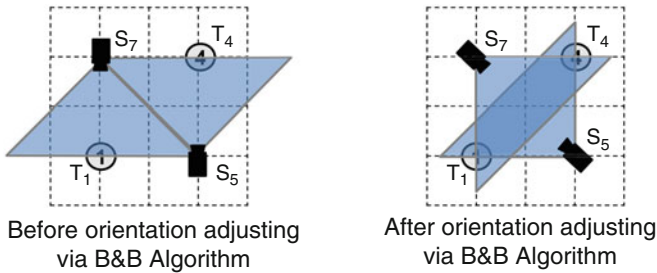


Fig. 11.5 B&B algorithm implementation sample

11.3.2 Hybridization of Branch and Bound with Genetic Algorithm

The GA is an effective algorithm for combinatorial optimization problems. It has population based search strategy that the solutions (individuals) are evolving to better solutions. The algorithms are chosen for designing more reliable WMSN topologies. The GA&HGA have same fitness function (f), representation type and B&B algorithm implementation with SA&HSA. It is obvious that in the representation, the gene of an individual is a sensor covering a target and its links to the b . The initial solution generating heuristic of the SA&HSA is also used for creating initial population for the GA based algorithms.

The proposed GA&HGA includes a tournament selection strategy for parental selection. A reliability-based crossover operator is implemented to choose two parents via selection strategy. The predefined f integrates connectivity and coverage simultaneously. Therefore, the proposed reliability-based crossover preserves connectivity of the children and trying to increase the coverage of more targets. The operator creates a gene pool from the genes of the parents and gives more chance to the more reliable sensors in the pool to be selected. Then a randomly selected

gene transfers to the child. The gene adding process continues until spending all TB . A crossover example is presented with a mutation example in Fig. 11.6. The neighbourhood function of the SA&HSA is established as the mutation operator of the GA&HGA.

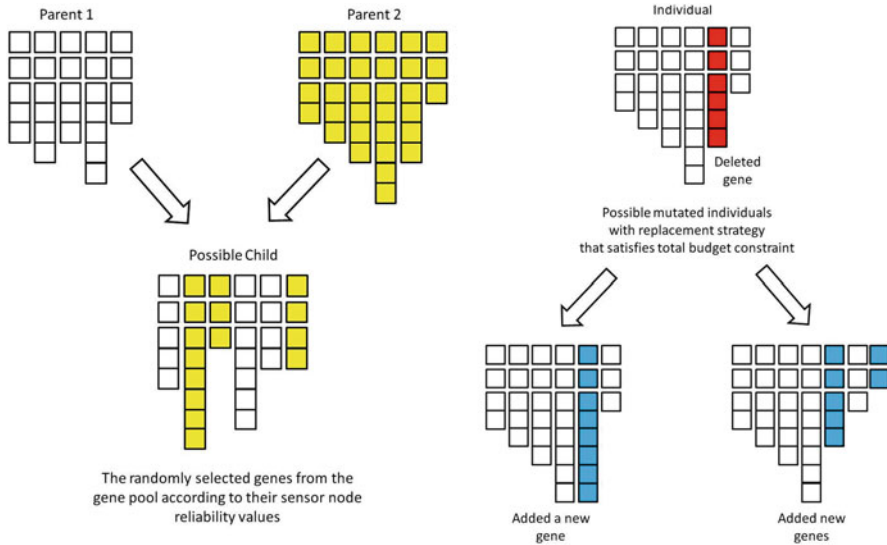


Fig. 11.6 Crossover and mutation operators

In the environmental selection, the new generation is replaced with the old generation. The population size (μ) does not change while creating new generations. Elitism strategy is also set into the algorithms and the selected best topologies of old generation is transferred automatically to the following generation. The stopping of the algorithm is to reach to the maximum number of generations ($maxgen$).

The sample size in tournament selection (tk), crossover probability (P_c), mutation probability (P_m), elitism rate (e), number of generated offsprings (λ), μ and $maxgen$ are tuned in experimental study.

11.4 Experimental Study

The details of the experiments are covered in this section. MATLAB 8.1 is used to code the algorithms. The experiments are performed on a Intel Core i7-3630QM computer which has 2.4 GHz CPU and 32 GB RAM. In the beginning, different scenarios and problem sets are created to test the performances of the algorithms. The sets are depicted in Table 11.2, which have different Ter , T , S and R properties. The sets are designed to examine the effects on the solutions of sensor types, relay node types, occlusions, Ter sizes, and TB amounts.

Table 11.2 Problem sets

#	<i>Ter</i> type	Size ($\times 10$ unit)	# of <i>T</i>	<i>ao</i> type ^a	<i>ao</i> density ^b	<i>no</i> type ^c	<i>sw</i> density ^d	<i>S</i> type	<i>c_j</i> (unit)	<i>r_j</i> ($\times 10$ unit)	<i>rel_j</i> ($\times 10$ unit)	<i>d_j</i> ($\times 10$ unit) (°)	<i>v_j</i> Type	<i>R</i> Type	<i>c_i</i> (unit)	<i>r_i</i> ($\times 10$ unit)	<i>rel_i</i>	<i>TB</i> (unit)
1	Small	6 × 6	3	0	0	0	0.9	1	8	1	0.9	2	90	1	5	3	0.9	40
2	Small	6 × 6	3	0	0	0	0.9	2	8	2	0.8	2	90	1	5	3	0.9	40
3	Small	6 × 6	3	0	0	0	0.9	3	8	2	0.9	1.5	90	1	5	3	0.9	40
4	Small	6 × 6	3	0	0	0	0.9	4	10	2	0.9	2	90	2	2	1	0.9	40
5	Small	6 × 6	3	0	0	0	0.9	4	10	2	0.9	2	90	3	2	3	0.7	40
6	Small	6 × 6	3	0	0	0	0.9	4	10	2	0.9	2	90	4	4	2	0.9	40
7	Small	6 × 6	3	0	0	0	0.9	4	10	2	0.9	2	90	5	4	3	0.8	40
8	Small	6 × 6	3	0	0	0	0.9	4	10	2	0.9	2	90	1	5	3	0.9	40
9	Small	6 × 6	3	0	0	0	0.9	4	10	2	0.9	2	90	1	5	3	0.9	50
10	Small	6 × 6	3	0	0	0	0.9	4	10	2	0.9	2	90	1	5	3	0.9	60
11	Moderate	8 × 8	4	0	0	0	0.9	4	10	2	0.9	2	90	1	5	3	0.9	70
12	Moderate	8 × 8	4	1	0.1	0	0.9	4	10	2	0.9	2	90	1	5	3	0.9	70
13	Moderate	8 × 8	4	0	0	1	0.9	4	10	2	0.9	2	90	1	5	3	0.9	70
14	Moderate	8 × 8	4	0	0	2	0.9	4	10	2	0.9	2	90	1	5	3	0.9	70
15	Moderate	8 × 8	4	0	0	3	0.9	4	10	2	0.9	2	90	1	5	3	0.9	70
16	Moderate	8 × 8	4	1	0.1	1	0.9	4	10	2	0.9	2	90	1	5	3	0.9	70
17	Moderate	8 × 8	4	1	0.1	2	0.9	4	10	2	0.9	2	90	1	5	3	0.9	70
18	Moderate	8 × 8	4	1	0.1	3	0.9	4	10	2	0.9	2	90	1	5	3	0.9	70
19	Moderate	8 × 8	4	0	0	0	0.9	4	10	2	0.9	2	90	1	5	3	0.9	80
20	Moderate	8 × 8	4	0	0	0	0.9	4	10	2	0.9	2	90	1	5	3	0.9	90
21	Large	10 × 10	5	0	0	0	0.9	4	10	2	0.9	2	90	1	5	3	0.9	100
22	Large	10 × 10	5	1	0.1	0	0.9	4	10	2	0.9	2	90	1	5	3	0.9	100
23	Large	10 × 10	5	0	0	1	0.9	4	10	2	0.9	2	90	1	5	3	0.9	100
24	Large	10 × 10	5	0	0	2	0.9	4	10	2	0.9	2	90	1	5	3	0.9	100
25	Large	10 × 10	5	0	0	3	0.9	4	10	2	0.9	2	90	1	5	3	0.9	100
26	Large	10 × 10	5	1	0.1	1	0.9	4	10	2	0.9	2	90	1	5	3	0.9	100
27	Large	10 × 10	5	1	0.1	2	0.9	4	10	2	0.9	2	90	1	5	3	0.9	100
28	Large	10 × 10	5	1	0.1	3	0.9	4	10	2	0.9	2	90	1	5	3	0.9	100
29	Large	10 × 10	5	0	0	0	0.9	4	10	2	0.9	2	90	1	5	3	0.9	110
30	Large	10 × 10	5	0	0	0	0.9	4	10	2	0.9	2	90	1	5	3	0.9	120

^a 0: No artificial occlusion on *Ter*. 1: There are artificial occlusions on *Ter*

^b The ratio of artificial occlusions on *Ter*

^c 0: No natural occlusion on *Ter*. 1: There are bushes on *Ter*. 2: There is woodland on *Ter*. 3: There is forest on *Ter*

^d The ratio of wholly safe candidate points on *Ter* (*sw* = 1)

11.4.1 Parameter Tuning

The parameters of SA and GA based algorithms are tuned with a small-sized problem set. The tuned parameters and the assigned values for the parameters of SA&HSA and GA&HGA are presented in Table 11.3.

Table 11.3 Tuned parameters of algorithms

SA&HSA parameter	Value
Initial temperature (T_0)	0.63°
Cooling parameters (α, β)	(0.8, 200)
Maximum iteration number ($maxitr$)	1000
GA&HGA parameter	Value
Population size (μ)	40
Sample size parameter in tournament selection (tk)	7
Crossover probability (P_c)	1
Mutation probability (P_m)	0.3
Elitism rate (e)	0.1
Number of generated offsprings (λ)	36
Maximum generation number ($maxgen$)	30

11.4.2 Performance Results

All problem sets are solved 10 times by the algorithms. Because the small and moderate-sized sets do not have any circumstances to use the B&B algorithm, the hybrid algorithms are run only for the large-sized sets. The results for the small and moderate-sized sets are summarized in Table 11.4 (highest results are bold). The CPU times of the algorithms change between 37 and 627 s. According to results, the GA has up to 6% better solutions than SA. Except problem set #13, the mean and the best results of GA are higher.

The performances of the proposed simple and hybrid algorithms on large-sized sets are provided in Table 11.5. The HGA outperforms the other algorithms in all cases and has up to 6% better solutions than the other algorithms. The CPU times of the algorithms increase in a range between 8 and 35 min.

For a reliability based problem, these improvements are enough to display the effectiveness of GA based algorithms to solve the predefined problem. Especially HGA shows good performance on large-sized sets with its hybrid content. It can be interpreted that the B&B algorithm can help to develop better algorithms to maintain better results for larger sized sets with increasing number of targets.

In Fig. 11.7, the effects of changing the S and R node types are depicted. The best results of the algorithms range up to 17%. The difference in the rel_j and c_j values

Table 11.4 Results for small and moderate-sized problem sets

#	SA			GA			#	SA			GA		
	Best	Mean	Worst	Best	Mean	Worst		Best	Mean	Worst	Best	Mean	Worst
1	0.762	0.736	0.700	0.774	0.745	0.700	11	0.862	0.835	0.759	0.875	0.860	0.852
2	0.612	0.586	0.457	0.637	0.620	0.604	12	0.863	0.841	0.818	0.866	0.851	0.832
3	0.768	0.709	0.608	0.787	0.737	0.707	13	0.817	0.767	0.743	0.812	0.783	0.740
4	0.763	0.718	0.672	0.767	0.735	0.685	14	0.534	0.509	0.497	0.573	0.540	0.518
5	0.772	0.723	0.677	0.792	0.769	0.738	15	0.333	0.294	0.227	0.342	0.321	0.299
6	0.704	0.609	0.566	0.711	0.694	0.679	16	0.783	0.750	0.674	0.792	0.771	0.747
7	0.696	0.638	0.519	0.759	0.680	0.646	17	0.515	0.478	0.442	0.528	0.509	0.487
8	0.745	0.707	0.581	0.755	0.739	0.724	18	0.328	0.311	0.291	0.339	0.316	0.299
9	0.806	0.794	0.767	0.845	0.810	0.787	19	0.888	0.861	0.847	0.889	0.872	0.854
10	0.845	0.818	0.786	0.856	0.842	0.819	20	0.887	0.868	0.841	0.889	0.886	0.882

Table 11.5 Results for large-sized problem sets

#	SA			GA			HSA			HGA		
	Best	Mean	Worst	Best	Mean	Worst	Best	Mean	Worst	Best	Mean	Worst
21	0.816	0.791	0.737	0.821	0.804	0.748	0.832	0.806	0.751	0.843	0.826	0.811
22	0.828	0.786	0.724	0.821	0.804	0.777	0.840	0.808	0.711	0.847	0.822	0.791
23	0.739	0.695	0.654	0.760	0.715	0.678	0.780	0.731	0.672	0.793	0.762	0.736
24	0.512	0.465	0.403	0.504	0.493	0.484	0.527	0.494	0.447	0.530	0.514	0.498
25	0.296	0.279	0.249	0.324	0.296	0.278	0.339	0.313	0.285	0.345	0.324	0.304
26	0.723	0.697	0.652	0.720	0.682	0.662	0.782	0.729	0.683	0.784	0.743	0.710
27	0.516	0.479	0.444	0.516	0.491	0.471	0.547	0.506	0.431	0.549	0.522	0.499
28	0.293	0.268	0.228	0.309	0.297	0.283	0.337	0.311	0.268	0.338	0.319	0.303
29	0.826	0.805	0.762	0.850	0.824	0.790	0.850	0.835	0.814	0.863	0.836	0.820
30	0.848	0.817	0.753	0.855	0.844	0.816	0.849	0.840	0.829	0.880	0.858	0.840

effected the results much more than the decrease of the other S properties (i.e. r_j and d_j). The rel_j is reduced from 0.9 to 0.8 and c_j is reduced from 10 to 8 in S type #2. Nonetheless, the gap in the c_l and r_l values in R types #4 and #5 also decreased the results.

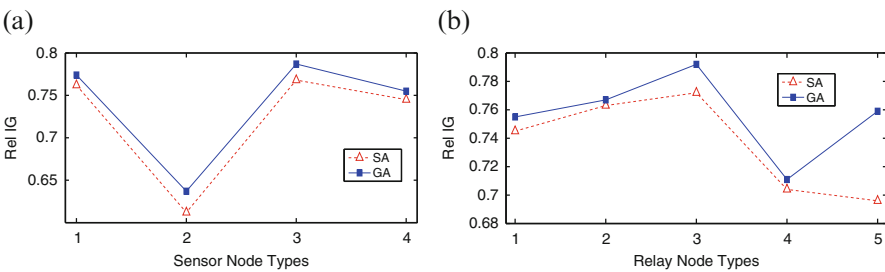


Fig. 11.7 The effects of S & R types. (a) The effects of S types. (b) The effects of R types

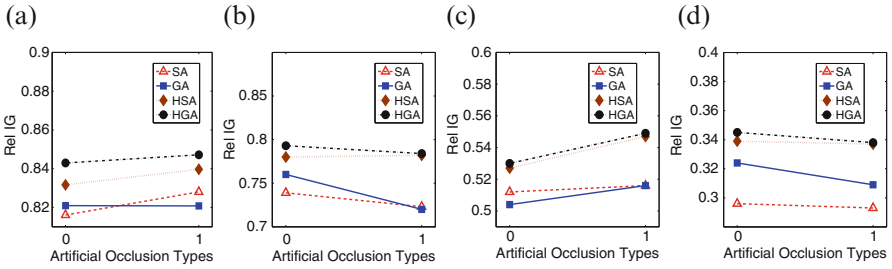


Fig. 11.8 The effects of *ao* types according to *no* types for large-sized sets. (a) *no* Type = 0. (b) *no* Type = 1. (c) *no* Type = 2. (d) *no* Type = 3

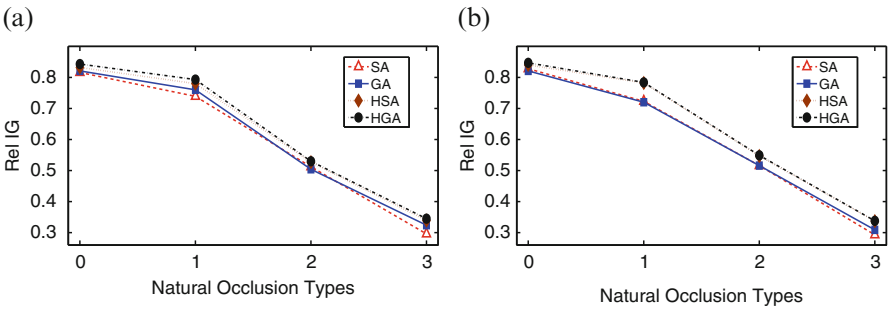


Fig. 11.9 The effects of *no* types according to *ao* types for large-sized sets. (a) *ao* Type = 0. (b) *ao* Type = 1

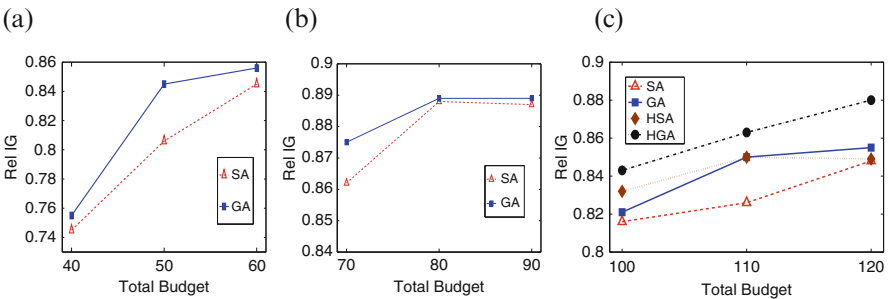


Fig. 11.10 The effects of *TB* amounts on *Ter* sizes. (a) *Ter* size = 6 × 6. (b) *Ter* size = 8 × 8. (c) *Ter* size = 10 × 10

In Fig. 11.8, the effects of adding *ao* to the scenarios are presented. The results of the algorithms decrease up to 4% when *ao* are placed on the *Ter*. In Fig. 11.9, the effects of considering three types of *no* to the results are displayed. The results decrease up to 8% when the *Ter* consists of bushes, 35% for woodland and 53% for the forest on the *Ter*, respectively. In Fig. 11.10, the effects of increasing the *TB* amounts are presented according to the *Ter* sizes. The results in 6 × 6 sized *Ter* increase roughly 10%, similarly the results in 8 × 8 approximately 10% and in 10 × 10 approximately 4% when 20 unit money is added to the *TB* amounts.

11.5 Conclusion

This chapter proposes novel hybridization methods, which involves an exact method and metaheuristics, to deploy sensor nodes for the maximization of WMSN reliability under a given budget constraint. A new reliability related metric, information gathering, is introduced to integrate connectivity and coverage issues in WMSN. In order to support more realistic scenarios, the proposed method considers the communication range, hardware reliability and cost of the nodes and terrain specific characteristics like occlusions, threat zones, and importance of the targets. Our hybridization methods combine B&B method with metaheuristics (SA and GA) so that HSA and HGA are employed to find the sensor node locations and the B&B approach is used to find the exact orientations of the cameras. Experimental study reveals that the hybrid methods can produce better results when it is compared to pure metaheuristics especially for large size problem sets.

References

1. I.F. Akyildiz, T. Melodia, K.R. Chowdhury, A survey on wireless multimedia sensor networks. *Comput. Netw.* **51**(4), 921–960 (2007)
2. A.L. Bajuelos, S. Canales, G. Hernández, A.M. Martins, Optimizing the minimum vertex guard set on simple polygons via a genetic algorithm. *WSEAS Trans. Inf. Sci. Appl.* **5**(11), 1584–1596 (2008)
3. M.O. Ball, Complexity of network reliability computations. *Networks* **10**(2), 153–165 (1980)
4. W.W. Bein, D. Bein, S. Malladi, Reliability and fault tolerance of coverage models for sensor networks. *Int. J. Sensor Netw.* **5**(4), 199–209 (2009)
5. A. Efrat, S. Har-Peled, Guarding galleries and terrains. *Inf. Process. Lett.* **100**(6), 238–245 (2006)
6. X. Han, X. Cao, E.L. Lloyd, C.C. Shen, Deploying directional sensor networks with guaranteed connectivity and coverage, in *5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'08)*, 2008, pp. 153–160
7. M. Marengoni, B. Draper, A. Hanson, R. Sitaraman, A system to place observers on a polyhedral terrain in polynomial time. *Image Vis. Comput.* **18**(10), 773–780 (2000)
8. A. Shrestha, L. Xing, H. Liu, Infrastructure communication reliability of wireless sensor networks, in *2nd International Symposium on Dependable Autonomic and Secure Computing (DASC 2006)*, 2006, pp. 250–257
9. A. Shrestha, L. Xing, H. Liu, Modeling and evaluating the reliability of wireless sensor networks, in *Reliability and Maintainability Symposium (RAMS'07)*, 2007, pp. 186–191
10. H. Topcuoglu, M. Ermis, I. Bekmezci, M. Sifyan, A new three-dimensional wireless multimedia sensor network simulation environment for connected coverage problems. *Simulation* **88**(1), 110–122 (2012)
11. J. Zhao, S.-C. Cheung, T. Nguyen, Optimal camera network configurations for visual tagging. *IEEE J. Sel. Top. Sign. Process.* **2**(4), 464–479 (2008)

Chapter 12

A Hybrid MCDM Approach for Supplier Selection with a Case Study

Hanane Assellaou, Brahim Ouhbi, and Bouchra Frikh

Abstract The supplier selection problem is one of the strategic decisions that have a significant impact on the performance of the supply chain. In this study, supplier selection problem of a well-known refining company in Africa is investigated and an integrated DEMATEL-ANP-TOPSIS methodology is used to select the best supplier providing the most customer satisfaction for the criteria determined. DEMATEL method is used in order to detect the cause and effect interaction among main criteria. The weights of criteria are calculated using Analytic Network Process approach and then the modified TOPSIS has been applied for the final selection. The supplier which is closest to the ideal solution and farthest from the negative ideal solution is selected as the best supplier.

Keywords Supplier selection • MCDM • DEMATEL • ANP • TOPSIS

12.1 Introduction

Since 1960s, supplier selection problem have been a focal point for many researchers. It is nowadays one of the critical topics in supply chain management. Besides, selection of suppliers is a complicated process by the facts that numerous criteria must be considered in the decision making process. Supplier selection and evaluation continues to be a key element in the industrial buying process and appears to be one of the major activities of the professional industrial [26].

H. Assellaou (✉) • B. Ouhbi
LM2I Laboratory, ENSAM-Meknes, Moulay Ismail University, Marjane II, BP 4024 BENI M'Hamed, Meknes, Morocco
e-mail: hanane.assellaou@gmail.com; ouhbib@yahoo.co.uk

B. Frikh
LTTI Laboratory, EST-Fès, Moulay Abdellah University, B.P. 1796 Atlas Fès, Fès, Morocco
e-mail: bfrikh@yahoo.com

It is a complex process to select a suitable supplier. Many factors should be taken into consideration when evaluating and selecting suppliers. In the literature, researchers have examined different criteria for the supplier selection problem. Dickson [7] identified 23 criteria based on a survey of 170 purchasing managers involved in various supplier selection problems, the study showed that supplier selection is a multi-criteria decision often involves the simultaneous consideration of several criteria such as price, delivery time and quality, and it is extremely difficult to find a perfect supplier. Weber et al. [28] analyzed 74 articles published between 1966 and 1990 dealing with this problem. Zhang et al. [30] compared Dickson and Weber study, and summarized new supplier selection criteria from the study of 49 articles from 1992 to 2003.

Extensive multi-criteria decision making approaches have been proposed for supplier selection, such as the Analytic Hierarchy Process (AHP), Analytic Network Process (ANP), Case-Based Reasoning (CBR), Data Envelopment Analysis (DEA), Genetic Algorithm (GA) and other methods. According to the review of 170 articles published during 2008–2012 about supplier selection which is presented by Chai et al. [3], there are about 40% of the papers use hybrid methods.

The combination of two methods provides pleasingly surprising results where an optimal mapping between the needs/ wants and the possible system alternatives can be achieved [13]. Furthermore, it helps to reduce complexity, increase the correctness, accuracy, and easiness of obtained results. In this study, in order to deal with the ranking abnormality and to reduce the number of handoffs, we conduct a hybrid method that combines DEMATEL, ANP and TOPSIS, this study can make better decisions in supplier selection.

Although Analytic Hierarchy Process (AHP) [20] is one of the most widely used Multi-criteria Decision Making (MCDM) methods which decomposes a problem into several levels that make up a hierarchy in which each decision element is supposed to be independent. AHP can only be employed in hierarchical decision models, however, many decision problems cannot be structured hierarchically and real world problems usually consist of dependence or feedback between elements. Analytic Network Process (ANP) (Saaty [21]) is a generalization of the AHP; it considers the dependence between the elements of the hierarchy. The ANP feedback approach replaces hierarchies with networks, and emphasizes interdependent relationships among various decision-making [17], also interdependencies among the decision criteria and permit more systematic analysis.

ANP is used very often in combination with other methods. ANP is used in combination with goal programming approach [12], and DEMATEL (Decision-Making Trial and Evaluation Laboratory) approach [29]. Lin et al. [16] used ANP and the Technique for Order Preference by Similarity to an Ideal Solution (TOPSIS) to establish real time purchasing environment. ANP and TOPSIS are used to calculate weight of criteria and rank suppliers. Demirtas and Üstün [5] applied ANP and Multi-Objective Programming (MOP) to a refrigerator manufacturing unit.

DEMATEL method was developed by Gabus and Fontela [8]. In recent years, the DEMATEL method that converts the mutual relationship between the criteria causes and effects from a complex system to an understandable structural model

has become very popular because it can visualize the structure of complicated causal relationships. DEMATEL method is introduced to build the structure of relationship map for clarifying the interrelations among criteria, as well as to visualize the causal relationship of criteria.

The TOPSIS (technique for order performance by similarity to ideal solution) method was originally proposed by Hwang and Yoon [10], some years later an improved version of this method, called the revised (or modified) TOPSIS, was proposed by Deng et al. [6] and has been widely used in the literature. TOPSIS simultaneously considers the distances to the ideal solution and negative ideal solution regarding each alternative and selects the most relative closeness to the ideal solution as the best alternative.

Petroleum refineries are very large industrial complexes that involve many different processing units and auxiliary facilities designed to produce physical and chemical changes in crude oil to convert it into everyday products like petrol, diesel, lubricating oil, fuel oil and bitumen. The dynamic development of the petroleum industry faces new challenges and directions such as increasing and more volatile energy prices. Petroleum refineries must select and maintain core suppliers to survive and succeed. This paper represents the case study of a refining company; we have identified some effective criteria which affect the process of supplier selection. DEMATEL method is used in order to detect the cause and effect interaction among main criteria. Then ANP method is implemented for calculating the weights of each criterion. Finally, TOPSIS method is applied for ranking suppliers based on data provided by the company. The result of this study gives the best possible solution of the supplier selection for the refining company.

This paper is organized as follows: Sect. 12.2 is a literature review that presents papers about supplier selection. Section 12.3 introduces the methodology used to select the best supplier. Section 12.4 describes the proposal model and an application case. Finally, Sect. 12.5 presents conclusions.

12.2 Related Works

There are several keywords associated with the supplier selection. The terms supplier selection and vendor selection are frequently used in the literature. Supplier selection and evaluation is one of the important stages in supply chain management which regards all the activities from the purchasing of raw material to final delivery of the product. It has received considerable attention for its significant effect toward successful logistic and supply chain management. Supplier selection problem is a kind of multiple criteria decision making problem which requires MCDM methods for solutions with high accuracy and numerous individual and integrated approaches were proposed to solve it.

Based on the literature reviews, a quick review of supplier selection models shows that many researches proposed methods based on ANP: Sarkis and Talluri [23] applied ANP to evaluate and select the best supplier with respect to organi-

zational factors and strategic performance metrics, which consist of 7 evaluating criteria. Sarkis [22] explored the applicability of ANP for decision making within the green supply chain. The author focused on the components and elements of green supply chain management and how they serve as a foundation for the decision framework. Gencer and Gurpinar [9] used ANP for the supplier selection problem, they implemented an ANP model in an electronic company to evaluate and select the best supplier with respect to various supplier evaluating criteria. Bayazit [1] proposed an ANP model to tackle the supplier selection problem with respect to ten evaluating criteria, which were classified into supplier performance and capability clusters.

However, ANP have two major disadvantages: the first one; it is difficult to provide the exact network structure among criteria, and different structures lead to different results. The second one is the formation of supermatrix; all criteria have to be pairwise compared with regard to all other criteria, which is difficult. Furthermore, the complexity increases exponentially with the number of criteria and their interdependencies, due both to the numbers of pairwise comparisons and to the dimensions of questionnaires sent to experts. However, by combining the ANP with the DEMATEL method the complexity of the problem can be reduced. The DEMATEL method does not require comparison of all pairs of criteria with respect to each individual criterion when establishing the inner dependencies of the criteria. Thus, the comparisons that need to be done are significantly reduced. The aim of DEMATEL is to convert the relation between elements, causal dimensions from a complex system to an understandable structural model.

The TOPSIS method developed by Hwang and Yoon [10] is a distance-based MCDM method that can be used for ranking alternatives, it has also found broad use in decision-making applications over the past few decades. On the other hand, TOPSIS method can reduce the difficulties of ANP in terms of presentation of interdependence between criteria and alternatives. It has been proved as one of the best methods in addressing the rank reversal issue.

In the literature, the DEMATEL, ANP and TOPSIS methodologies have been combined for some realized applications. Chen and Chen [4] applied decision-making trial and evaluation laboratory (DEMATEL), fuzzy ANP, and TOPSIS to develop a new innovation support system for Taiwanese higher education. Buyukozkan and Cifci [2] proposed a hybrid fuzzy MCDM model which combines the DEMATEL, ANP, and TOPSIS in a fuzzy context for green supplier evaluation. Lin et al. [15] evaluated vehicle telematics system by using DEMATEL, ANP, and TOPSIS techniques with dependence and feedback. Ju and Wang [11] proposed a framework combining the ANP method, the DEMATEL technique, and 2-tuple linguistic TOPSIS (TL-TOPSIS) method to solve the emergency alternative evaluation and selection problem for a practical example of urban fire emergency alternative selection. Vinodh et al. [27] used the combination of DEMATEL, ANP and TOPSIS in order to enhance the effectiveness of agile concept selection. The study is aimed at selecting the best concept design of an automobile component.

These three approaches used by several authors are workable. Because by applying these theories, it can be easy to discover things inside the complex problem.

However, although these kinds of combined works have increased in the recent years, there are a very few studies that combined DEMATEL, ANP, and TOPSIS for supplier selection.

The selection of supplier problem is still challenging, selecting the right supplier becomes a critical activity within a company and consequently affects its efficiency and profitability. Since supplier evaluation and selection is a current topic and due to his strategic importance, important research is being done to cope with this problem.

12.3 Methodology

In the literature, many techniques have been developed to solve the supplier selection problem, and all techniques have to use criteria to rate suppliers. Most of them are based on the additive concept along with the independence assumption, but each individual criterion is not always completely independent [29]. Applicability of the DEMATEL, ANP and TOPSIS methods for solving various problems is already proven. In the literature there are papers using these methods, but there is only few studies that combines these three methods together to solve the problem of supplier selection. Thereby, this paper proposes the integrated DEMATEL-ANP-TOPSIS approach that can solve complex problems and adequately consider all the relationships between the factors, criteria and alternatives

To take into account the interactions among elements, the ANP and DEMATEL were proposed. ANP, widely applied in decision making, is more accurate and feasible under interdependent situations. This method allows assessing the consistency of the judgments and facilitates the process of assigning weights by splitting up the problem into smaller parts, appropriate for more detailed analysis. On the other hand, DEMATEL method employed as a supportive tool for ANP is introduced to build the structure of relationship map for clarifying the interrelations among criteria, as well as to visualize the causal relationship of criteria through a causal diagram. ANP method is a mathematical theory that can solve all kinds of dependencies, but it doesn't solve the problem completely because using ANP to solve MCDM problems has different influence levels among criteria based on Network Relationship Maps (NRM) [24]. To avoid calculation and additional pairwise comparisons of ANP, TOPSIS is used to rank the alternatives. It can reduce the difficulties of ANP in terms of presentation of interdependence between criteria and alternatives.

In this paper, DEMATEL is employed to identify the relationship network among the criteria, then ANP method is used to derive weights that account for component interaction. Finally, the TOPSIS method (Technique for Order of Preference by Similarity to Ideal Solution) is introduced to rank alternatives; it is based on the relative similarity to the ideal solution, which avoids from the situation of having same similarity to both ideal and negative ideal solutions. Combining three MCDM methods (DEMATEL, ANP, and TOPSIS) helps to determine the final score of each supplier.

The choice for an integrated DEMATEL-ANP-TOPSIS based framework proposed in this paper is justified by several reasons:

- DEMATEL helps researchers better understand the nature of the problem.
- DEMATEL involves indirect relations into a compromised cause and effect model.
- It is an effective approach to identify key factor criteria.
- It can prioritize the criteria based on the type of relationships and severity of influences they have on one another.
- The ANP model provides a looser network structure that makes possible the representation of any decision problem.
- ANP is capable of handling feedback and interdependencies.
- ANP depicts the dependence and influences of the factors involved to the goal or higher-level performance objective.
- TOPSIS has also found broad use in decision-making applications over the past few decades.
- TOPSIS method provides a sound logic that represents the rationale of human choice [25], a scalar value that accounts for the best and worst alternative choices simultaneously, and a simple computation process that can easily be programmed into a spreadsheet.
- The concept of TOPSIS is understandable, and the computation involved is uncomplicated.
- TOPSIS can rank the alternative locations based on their overall performance, since it may identify the best solution that is closest to the positive ideal solution and farthest from negative ideal solution.
- A relative advantage of TOPSIS is the ability to identify the best alternative quickly [19], it is not so complicated for the managers as some other methods which demand additional knowledge.

In the next, we present a brief introduction on the DEMATEL, ANP, and TOPSIS methods.

12.3.1 DEMATEL Method

The DEMATEL method [8] is especially pragmatic to visualize the structure of complicated causal relationships with matrices or diagraphs. The aim of DEMATEL is to convert the relation between elements, causal dimensions from a complex system to an understandable structural model [14]. The steps of DEMATEL technique are explained below:

Step 1 Generating the direct-relation matrix: An evaluation scale of 0, 1, 2, 3, and 4 is used for influential comparison where 0 represents an influence while 4 represents every high influence. A group of experts is asked to make pairwise comparisons between criteria. To compound all opinions from K experts, the

direct-relation matrix A is calculated using Eq. (12.1) by averaging each expert scores.

$$a_{ij} = \frac{1}{K} \sum_{k=1}^K x_{ij}^k \quad (12.1)$$

where x_{ij}^k is the score given by the k th expert indicating the influential level that the factor i has on factor j .

Step 2 Normalizing the direct-relation matrix: The normalized direct-relation matrix M can be obtained by normalizing A using Eqs. (12.2) and (12.3).

$$M = p.A \quad (12.2)$$

$$p = \min\left(\frac{1}{\max_{1 \leq i \leq n} \sum_{j=1}^n a_{ij}}, \frac{1}{\max_{1 \leq j \leq n} \sum_{i=1}^n a_{ij}}\right) \quad (12.3)$$

Step 3 Calculating the total-relation matrix: The total-relation matrix T can be obtained by using Eq. (12.4), where I denotes the identity matrix.

$$T = M(I - M)^{-1} \quad (12.4)$$

where $T = [t_{ij}]_{n \times n}$, $i, j = 1, 2, \dots, n$

Step 4 Compute the dispatcher group and receiver group: The vectors $c = [c_i]_{n \times 1}$ and $d = [d_j]_{1 \times n}$ represent respectively the sum of rows and the sum of columns of matrix T respectively (Eqs. (12.5) and (12.6)). $c + d$ value indicates the degree of importance that the corresponding criterion plays in the entire system. The criterion having greater value of $c + d$ has more interrelationships with other criteria. On the other hand, criteria having positive values of $c - d$ are on the cause group and dispatches effects to the other criteria. On the contrary, criteria having negative values of $c - d$ are on the effect group and receive effects from the other criteria.

$$c_i = \sum_{j=1}^n t_{ij}, \quad \forall i \in \{1, \dots, n\} \quad (12.5)$$

$$d_j = \sum_{i=1}^n t_{ij}, \quad \forall j \in \{1, \dots, n\} \quad (12.6)$$

Step 5 Set up a threshold value.

12.3.2 ANP Method

The Analytic Network Process (ANP) was developed by Thomas Saaty [21], in his work on multi criteria decision making. The ANP has its own advantages and has produced ideal results in various fields. It applies network structures with

dependence and feedback, among the criteria, to complex decision making. Pairwise comparisons are made based on Saaty [20] 1–9 scale. In the application of ANP, software like, Ecnnet, Super Decision or mathematical programs like Excel, Maple, Mathematica can be used. ANP has four stages; the first stage is network structure formation. The relationships captured in this step constitute both within clusters and between clusters. The second stage is the formation of pairwise comparisons and obtaining local priority values. Without assuming the interdependence between criteria, pairwise comparisons are performed and the priority value of each criterion in the network structure is obtained. After assigning the values of pairwise comparisons in the comparison matrix, local priority vector is calculated from eigenvector, which is calculated using Eq. (12.7). All responses are gathered based on Saaty scale.

Let A , W_1 and λ_{\max} be respectively a pairwise comparison matrix, eigenvector and eigenvalue, respectively in Eq. (12.7),

$$\lambda_{\max} \cdot W_1 = A \cdot W_1 \quad (12.7)$$

All obtained vectors are further normalized to represent the local priority vector W_2 . Indeed the normalized pairwise comparison matrix B is obtained. The matrix B consists of b_{ij} values, which are calculated using Eq. (12.8).

$$b_{ij} = \frac{a_{ij}}{\sum_{i=1}^n a_{ij}} \quad (12.8)$$

The eigenvector (W_2) is obtained by using Eq. (12.9)

$$W_2 = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}, w_i = \frac{\sum_{i=1}^n b_{ij}}{n} \text{ for } i = 1, 3, \dots, n \quad (12.9)$$

After that, λ_{\max} is obtained using Eq. (12.10) and the consistency property is checked after performing the Eqs. (12.11) and (12.12). Let CI, RI and CR denote consistency indicator, random indicator and consistency ratio, respectively. RI is obtained from a standard random index table showing the random index values for different number of criteria regarded. Consistency ratio must be smaller than 0.10 [20].

$$W' = A \cdot W_2 = \begin{bmatrix} w'_1 \\ w'_2 \\ \vdots \\ w'_n \end{bmatrix}, \text{ and } \lambda_{\max} = \frac{1}{n} \left(\frac{w'_1}{w_1} + \frac{w'_2}{w_2} + \dots + \frac{w'_n}{w_n} \right) \text{ for } i = 1, 3, \dots, n \quad (12.10)$$

$$CI = \frac{\lambda_{\max} - n}{n - 1} \quad (12.11)$$

$$CR = \frac{CI}{RI} \quad (12.12)$$

In the next stage, the interdependence between the criteria is considered. The super matrix is obtained by locating the local priority vectors, generated by the pairwise comparison matrix, on convenient columns. In general, the sum of one column in super matrix is greater than 1. Unless a stochastic super matrix is obtained, the cluster is weighted and normalization is performed to obtain a stochastic matrix where the sum of column values is 1. This newly obtained super matrix C is often called as weighted super matrix. The last stage is to obtain the interdependence priorities of the criteria by synthesizing the results from the previous step as follows:

$$W_c = CW_2^T \quad (12.13)$$

12.3.3 TOPSIS Method

TOPSIS method is a multiple criteria method to identify solutions from a finite set of alternatives, it was originally proposed by Hwang and Yoon [10], some years later an improved version of this method, called the revised TOPSIS, was proposed by Deng et al. [6]. TOPSIS is a widely used MCDM technique because of its logic and its programmable computation procedure [18]. The concept of TOPSIS is that an alternative which is closest to the ideal solution and farthest from the negative ideal solution in a multi-dimensional computing space is the optimal choice. Therefore, the preference order of alternatives is yielded through comparing Euclidean distances. Supposed that there are m ($m > 1$) alternatives A_1, \dots, A_m , the TOPSIS process is carried out as follows [26]:

Step 1 Build a decision matrix (D) with values of criteria

$$D = \begin{bmatrix} X_{11} & \cdots & X_{1j} & X_{1n} \\ \vdots & & \ddots & \vdots \\ X_{m1} & \cdots & X_{mj} & X_{mn} \end{bmatrix}$$

Step 2 Normalize the decision matrix (D) through the following equation:

$$r_{ij} = \frac{X_{ij}}{\sqrt{\sum_{i=1}^n X_{ij}^2}} \quad i = 1, \dots, m; j = 1, \dots, n \quad (12.14)$$

Step 3 Determine the ideal solution and negative solution through the following equation:

$$R^+ = \{r_1^+ \cdots r_n^+\} = \left\{ \max_i \{r_{ij} \mid j \in J\}, \min_i \{r_{ij} \mid j \in J'\} \right\} \quad (12.15)$$

$$R^- = \{r_1^- \cdots r_n^-\} = \left\{ \min_i \{r_{ij} \mid j \in J\}, \max_i \{r_{ij} \mid j \in J'\} \right\} \quad (12.16)$$

where J is associated with the benefit criteria and J' is associated with the cost criteria.

Step 4 Compute the distance between each alternative and the ideal solution, and each alternative and negative solution: The separation measure D_i^+ of each alternative from the ideal solution is given as,

$$D_i^+ = \sqrt{\sum_{j=1}^n w_j (r_{ij} - r_j^+)^2} \quad (12.17)$$

Where w_j represents the weight of the j_{th} criterion. Similarly, the separation measure D_i^- of each alternative from the negative solution is given as follows:

$$D_i^- = \sqrt{\sum_{j=1}^n w_j (r_{ij} - r_j^-)^2} \quad (12.18)$$

Step 5 Calculate the relative closeness to the ideal solution of each alternative:

$$C_i^* = \frac{D_i^-}{D_i^- + D_i^+} \quad (12.19)$$

Step 6 Rank the order of alternatives and choose the best supplier.

12.4 Proposed Methodology and Application Case

The contribution of the current study lies in the practical implementation of the integration of the DEMATEL, ANP, and TOPSIS methods that will enable the proposed framework to be used by experts in a real industry for determining the appropriate suppliers.

The proposed methodology for the supplier selection problem consists of four basic stages: (1) identify the criteria, (2) application of DEMATEL method (3) ANP computations, (4) evaluation of alternatives with TOPSIS and determination of the final rank. The proposed model is applied to a real problem; the industrial data is collected from one of top most refining company. The case pertains to decision related to supplier selection in Moroccan corporation of the refining industry (Société Anonyme Marocaine de l'Industrie du Raffinage) (SAMIR). The case company

specialized in the refining of petroleum products, is located in the city of Mohammedia. It is considered one of the giants in the field of refining in Africa. In this study, there are three products and nine suppliers proposed by the company for selection, the aim of our specific study is to assess possible alternative supplier solutions and to help the decision-makers accordingly in terms of user requirements.

12.4.1 Identification of Necessary Criteria for Supplier Selection

The first step of our proposed methodology is the formulation of necessary criteria for supplier selection. Supplier selection decisions are complicated by the fact that various criteria must be considered in decisions making process. The analysis of such criteria for the selection, measuring the performances of potential suppliers and the introduction of new categories of selection criteria following the market evolutions have been the focus of many researchers and purchasing practitioners since 1960s. Based on expert opinions as well as the result of previous studies, past experience and the background of the expert team, this study selects 7 important criteria for selection of best supplier. The list of criteria determined are listed and briefly described as below:

Quality: it is the most important requirement of an organization to succeed in a competitive market place. It refers to a supplier product quality.

Flexibility: The ability of a supplier to accommodate changes in the enterprise production plans such as the ability to adjust product volume, the ability to customize product as demanded by the buyer, the ability to adjust manufacturing process, and the ability to fill emergency orders with required amount in a required time.

Price: it is related to the acquisition such as the purchase cost of materials, the transportation cost, and handling and package cost.

Technology: it refers to the presence of a technological system that can facilitate technology development of products and involvement to formulating new products. This criterion is related also to the improvement effort in products and processes, and the problem solving capability.

Delivery: The ability of the supplier to follow the predefined delivery schedule, so how well a supplier succeeds in delivering goods according to schedule?

Responsiveness: referring to all communication efforts from supplier towards the buyer, the after sales service, and support provided by a supplier.

Credit risk: refers to the probability of loss due to a borrower failure to make payments on any type of debt. Given the implications and probabilities of supplier defaults, the need to integrate a systematic assessment of credit risk into the supplier-selection process is clear.

12.4.2 Applying DEMATEL for Constructing the Interdependence Relationship Network

The questionnaires from three experts are used to determine the level of relationship among criteria. The specific calculation processes are described as follows. The evaluations of the experts are obtained and then averages of numbers are calculated using Eq. (12.1) in order to form initial direct-relation matrix (see Table 12.1).

Table 12.1 The average initial direct-relation matrix A

Criteria	Quality	Delivery	Flexility	Responsiveness	Technology	Credit risk	Price
Quality	0	1	1667	2667	3667	1667	4
Delivery	1667	0	2	2333	1667	1667	3333
flexibility	2	2	0	2667	2667	1667	2667
responsiveness	3667	2333	2667	0	2333	2333	3
Technology	3667	2667	1333	3	0	1667	4
Credit risk	2333	2667	3	3	2667	0	3
Price	4	2667	2333	3	2667	3667	0

The normalized direct-relation matrix is obtained using Eqs. (12.2) and (12.3). After calculating the normalized direct-relation matrix, the total-relation matrix is obtained using Eq. (12.4). The total-relation matrix is shown in Table 12.2. The threshold value is determined as 0.5 by the experts.

Table 12.2 The total-relation matrix

Criteria	Quality	Delivery	Flexility	Responsiveness	Technology	Credit risk	Price
Quality	0.794	0.656	0.668	0.855	0.875	0.682	1.050
Delivery	0.762	0.518	0.605	0.738	0.689	0.599	0.9
Flexibility	0.820	0.649	0.536	0.793	0.771	0.629	0.921
responsiveness	1.007	0.751	0.752	0.776	0.867	0.747	1.068
Technology	1.022	0.774	0.703	0.927	0.763	0.730	1.122
Credit risk	0.949	0.768	0.768	0.902	0.877	0.633	1.065
Price	1.109	0.835	0.807	1.001	0.962	0.870	1.027

The total influences given to and received by each criterion are given in Table 12.3. The factor having greater value of $c + d$ has more interrelationships with other criteria. On the other hand, criteria having positive values of $c - d$ are on the cause group and dispatches effects to the other criteria. On the contrary, criteria having negative values of $c - d$ are on the effect group and receive effects from the other criteria. If $c_i - d_i > 0$, it means that the degree of affect on the others is stronger than the degree it is affected.

Table 12.3 The sum of influences given and received on criteria

Criteria	c_i	d_i	$c_i - d_i$	$c_i + d_i$
Quality	5.3534	5.5814	-0.228	10.9348
Delivery	4.1156	4.8098	-0.6942	8.9254
Flexibility	4.0321	5.1186	-1.0865	9.1507
responsiveness	4.991	5.9675	-0.9765	10.9585
Technology	4.8426	6.0406	-1.198	10.8832
Credit risk	4.0191	5.961	-1.9419	9.9801
Price	6.1251	6.6101	-0.485	12.7352
$\alpha = 0.5$				

The resulting NRM is given in Fig. 12.1 from which the interrelationship among the seven criteria can be determined. According to experts opinions through DEMATEL analysis, almost all criteria are mutually interrelated. This can be seen from the double-sided arrows among the seven criteria in Fig. 12.1.

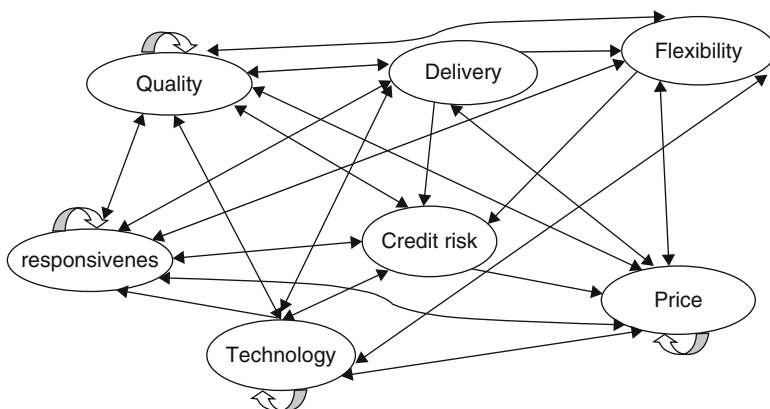


Fig. 12.1 Network relationship map of criteria for the supplier selection problem

12.4.3 The Weights of Criteria Calculation

After forming the decision network for the problem, the weights of the criteria to be used in evaluation process are calculated by using ANP method. Firstly, the decision makers are asked to evaluate all proposed criteria pairwise without assuming the interdependence between criteria. The result is presented in Table 12.4. The normalized eigenvector is $W_2 = (0.264, 0.134, 0.168, 0.137, 0.111, 0.032, 0.150)$ which presents the related local priority of these criteria.

In addition, we considered the dependence among the selection criteria. The decision makers examined the impact of all criteria on each other by using pairwise comparisons. Various pairwise comparison matrices are constructed for each of the

Table 12.4 The pairwise comparison matrix for criteria

Criteria	Quality	Flex.	Price	Technology	Delivery	Resp.	Credit risk	e-vectors
Quality	1	4	3	3	3	5	3	0.264
Flexibility	1/4	1	1/3	5	2	3	1/5	0.134
Price	1/3	3	1	1/2	1/3	5	7	0.168
Technology	1/3	1/5	2	1	3	3	3	0.137
Delivery	1/3	1/2	3	1/3	1	5	1/5	0.111
Responsiveness	1/5	1/3	1/5	1/3	1/5	1	1/3	0.032
Credit risk	1/3	5	1/7	1/3	5	3	1	0.150

criterion. These pairwise comparison matrices are needed to identify the relative impacts of criteria interdependent relationships. In total, seven pairwise comparison matrices were developed. The normalized eigenvectors for these matrices are calculated and shown as seven columns in Table 12.5.

Table 12.5 The degree of relative impact for evaluation criteria

Criteria	Quality	Delivery	Flexibility	Responsiveness	Technology	Credit risk	Price
Quality	1	0.327	0.351	0.310	0.372	0.298	0.322
Delivery	0.245	1	0.272	0.219	0.219	0.211	0.209
Flexibility	0.160	0.257	1	0.181	0.142	0.181	0.224
Responsiveness	0.110	0.170	0.131	1	0.115	0.149	0.113
Technology	0.096	0.116	0.095	0.127	1	0.088	0.078
Credit risk	0.038	0.077	0.110	0.098	0.083	1	0.051
Price	0.348	0.051	0.039	0.063	0.067	0.070	1

To get relative importance weights, we first normalize the weights in each column to sum up to one (i.e. make the matrix column stochastic). Then we complete this task by dividing every element in a column by the sum of that column. To obtain the converged set of weights, we raise the super-matrix C to a large power. In our case, convergence of C occurred when we raised the super-matrix to the 32nd power (see Table 12.6).

Table 12.6 The degree of relative impact for evaluation criteria

	Quality	Delivery	Flexibility	Responsiveness	Technology	Credit risk	Price
Quality	0.248	0.248	0.248	0.248	0.248	0.248	0.248
Delivery	0.190	0.190	0.190	0.190	0.190	0.190	0.190
Flexibility	0.162	0.162	0.162	0.162	0.162	0.162	0.162
Responsiveness	0.115	0.115	0.115	0.115	0.115	0.115	0.115
Technology	0.091	0.091	0.091	0.091	0.091	0.091	0.091
Credit risk	0.067	0.067	0.067	0.067	0.067	0.067	0.067
Price	0.120	0.120	0.120	0.120	0.120	0.120	0.120

Then, we obtain the interdependence priorities of the criteria by synthesizing the results from the previous step as follows:

$$W_c = CW_2^T.$$

Thus, the weights of the evaluation criteria can be determined

$$W_c = \begin{bmatrix} 0.248 & 0.248 & 0.248 & 0.248 & 0.248 & 0.248 & 0.248 \\ 0.190 & 0.190 & 0.190 & 0.190 & 0.190 & 0.190 & 0.190 \\ 0.162 & 0.162 & 0.162 & 0.162 & 0.162 & 0.162 & 0.162 \\ 0.115 & 0.115 & 0.115 & 0.115 & 0.115 & 0.115 & 0.115 \\ 0.091 & 0.091 & 0.091 & 0.091 & 0.091 & 0.091 & 0.091 \\ 0.067 & 0.067 & 0.067 & 0.067 & 0.067 & 0.067 & 0.067 \\ 0.120 & 0.120 & 0.120 & 0.120 & 0.120 & 0.120 & 0.120 \end{bmatrix} \begin{bmatrix} 0.264 \\ 0.111 \\ 0.134 \\ 0.032 \\ 0.137 \\ 0.150 \\ 0.168 \end{bmatrix}$$

$$W_c = (0.248, 0.190, 0.162, 0.115, 0.091, 0.067, 0.120)$$

W_c (**Quality** (C_1), **delivery** (C_2), **flexibility** (C_3), responsiveness (C_4), technology (C_5), credit risk (C_6), Price (C_7)) According to the vector W_c , quality, delivery and flexibility are three of the most important factors related to the evaluation supplier selection process.

12.4.4 Application of TOPSIS in Alternatives Ranking

By contacting the company, the warehouse manager and a production engineer suggested us three different products, each product has three suppliers. Process chemicals are the basis of chemical mixtures and chemical solution (acid, base, alcohol, ester...) and are used in the refining industry to prevent corrosion, fouling and damage of equipment. Mechanical seals are mechanical parts manufactured according to planes and unique to each equipment (pump or compressor), and are used for pumps and allow a safely operation of rotating machines. Protective equipment are the set of personal protective equipment i.e. (helmet, blue, shoes, goggles, gloves...) and are binding in the enclosure of the refinery for any individual. In order to protect respondent confidentiality, supplier names used throughout the article are given in the form of letters X, Y, Z respectively for the three products (see Table 12.7).

Table 12.7 The degree of relative impact for evaluation criteria

Product	Suppliers
P_1 : Process chemicals used for refining processes	X_1 X_2 X_3
P_2 : mechanical seals (spares) used for rotating machines (pumps)	Y_1 Y_2 Y_3
P_3 : protective equipments used for industrial safety	Z_1 Z_2 Z_3

At this stage of the decision procedure, the team members were asked to establish the decision matrix by comparing alternatives under each of the criteria separately. In addition, the evaluators were asked to provide a set of crisp values within a range

from 1 to 10 that represents the performance of each supplier with respect to each criterion. In this study, we have 3 alternatives for each product. D_1 , D_2 and D_3 are respectively the decision matrix of the products P_1 , P_2 and P_3 :

$$D_1 = \begin{matrix} & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 \\ \begin{matrix} X_1 \\ X_2 \\ X_3 \end{matrix} & \begin{pmatrix} 7 & 7 & 6 & 8 & 8 & 7 & 8 \\ 7 & 5 & 5 & 6 & 7 & 5 & 8 \\ 7 & 7 & 6 & 6 & 7 & 6 & 6 \end{pmatrix} \end{matrix}, \quad D_2 = \begin{matrix} & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 \\ \begin{matrix} Y_1 \\ Y_2 \\ Y_3 \end{matrix} & \begin{pmatrix} 8 & 7 & 7 & 6 & 6 & 5 & 4 \\ 8 & 2 & 3 & 4 & 6 & 1 & 2 \\ 8 & 5 & 6 & 5 & 6 & 3 & 6 \end{pmatrix} \end{matrix},$$

$$D_3 = \begin{matrix} & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 \\ \begin{matrix} Z_1 \\ Z_2 \\ Z_3 \end{matrix} & \begin{pmatrix} 4 & 3 & 3 & 5 & 7 & 4 & 2 \\ 5 & 6 & 5 & 5 & 7 & 5 & 5 \\ 8 & 7 & 6 & 7 & 7 & 6 & 7 \end{pmatrix} \end{matrix},$$

where the weights of selected criteria (quality (C_1), delivery (C_2), flexibility (C_3), responsiveness (C_4), technology (C_5), credit risk (C_6), Price (C_7)) are respectively (0.248, 0.190, 0.162, 0.115, 0.091, 0.067, 0.120) as obtained in the ANP analysis. By using Eqs. (12.17) and (12.18), the computed distances of each supplier from ideal solution (D_i^+) and negative solution (D_i^-) are presented in Table 12.8 respectively. Based on their relative closeness to the ideal solution (Table 12.9) obtained by using Eq. (12.19), the final step of the TOPSIS method consists of ranking supplier alternatives. In this case, the results show that, for product P_1 , X_1 is the best choice among alternatives, with a performance value of 0.608; X_3 and X_2 have been ranked second and third, with 0.584 and 0.281 as the performance values, respectively. For product P_2 , Y_1 is the best choice among alternatives, with a performance value of 0.620; Y_3 and Y_2 have been ranked second and third, with 0.479 and 0.450 as the performance values, respectively and finally for product P_3 , Z_3 is the best choice, with a performance value of 0.603; Z_2 and Z_1 have been ranked second and third, with 0.470 and 0.396 as the performance values respectively.

Table 12.8 The separation measure D_i^+ and D_i^- of each alternative from the ideal and the negative solution respectively

	D_1^+	D_2^+	D_3^+	D_1^-	D_2^-	D_3^-
P_1	0.073	0.126	0.126	0.113	0.049	0.106
P_2	0.197	0.310	0.235	0.324	0.254	0.217
P_3	0.311	0.215	0.204	0.204	0.191	0.311

Table 12.9 The relative closeness to the ideal solution of each alternative

Alternatives (P_1)	C_i^*	Alternatives (P_2)	C_i^*	Alternatives (P_3)	C_i^*
X_1	0.608	Y_1	0.620	Z_1	0.396
X_2	0.281	Y_2	0.450	Z_2	0.470
X_3	0.584	Y_3	0.479	Z_3	0.603

In this empirical study, we compared the results obtained by the used approach with the choice of the company; we found perfect concordance which shows the effectiveness of the combination of DEMATEL, ANP and TOPSIS methods.

12.5 Conclusion

Supplier selection is difficult given the qualitative and quantitative criteria. Since selecting the best supplier involves complex decision variables, it is considered to be a multi criteria decision problem. In this context, a case study was performed to decide the best supplier for three products for a Moroccan company of refining industry. The DEMATEL method, ANP, and TOPSIS are integrated in this paper to help a refining company to effectively select the best suppliers.

In this paper, nine suppliers are evaluated with regard to seven criteria; the criteria used for the evaluation were quality, delivery, flexibility, responsiveness, technology, credit risk and Price.

According to the closeness coefficient, one can determine not only the ranking order but also the assessment status of all possible suppliers. It has been shown that X_1 is the suitable supplier among the given alternatives for process chemicals, Y_1 for mechanical seals and finally Z_3 for protective equipments.

As a result of the empirical study, we found that the DEMATEL-ANP-TOPSIS methodology was a practical and efficient tool for ranking candidate suppliers in terms of their overall performance with respect to multiple criteria. Using an integrated DEMATEL, ANP TOPSIS for supplier selection and evaluation problem can reduce ambiguities and vagueness that are inherent in the field of supplier selection management decision problems.

The case company takes this result for comparison. After interviewing the managers, they point out that the selected supplier for each product is better based on their survey and annual performance evaluation data. Another important finding is that the proposed approach is more reflecting the relation of how the selection criteria affect the selected suppliers and at the same time what is more important for the suppliers among the selection criteria.

This paper moves us one step closer to the usage of the integrated DEMATEL, ANP, and TOPSIS in real world situations. In this study, DEMATEL, ANP and TOPSIS work well together if there are no random (random demand, variable cost..). Recently, much attention has been given to stochastic demand due to uncertainty in the real world. The main objective of the future work is to propose an approach for supplier selection when the buyer is faced with random demands and variable cost.

References

1. O. Bayazit, Use of analytic network process in vendor selection decisions. *Benchmarking Int. J.* **13**(5), 566–579 (2006)
2. G. Buyukozkan, G. Cifci, A novel hybrid MCDM approach based on fuzzy DEMATEL, fuzzy ANP and fuzzy TOPSIS to evaluate green suppliers. *Expert Syst. Appl.* **39**(3), 3000–3011 (2012)
3. J. Chai, J.N.K. Liu, E.W.T. Ngai, Application of decision-making techniques in supplier selection: a systematic review of literature. *Expert Syst. Appl.* **40**, 3872–3885 (2013)
4. J.K. Chen, I.S. Chen, Using a novel conjunctive MCDM approach based on DEMATEL, fuzzy ANP, and TOPSIS as an innovation support system for Taiwanese higher education. *Expert Syst. Appl.* **37**(3), 1981–1990 (2010)
5. E.A. Demirtas, Ö. Üstün, An integrated multi objective decision making process for supplier selection and order allocation. *Omega* **36**, 76–90 (2008)
6. H. Deng, J.W. Robert, C.H. Yeh, Inter-company comparison using modified TOPSIS with objective weight. *Comput. Oper. Res.* **27**(10), 963–973 (2000)
7. G.W. Dickson, An analysis of supplier selection system and decision. *J. Purch.* **2**(1), 5–17 (1966)
8. A. Gabus, E. Fontela, *World Problems an Invitation to Further Thought Within the Framework of DEMATEL* (Battelle Geneva Research Centre, Geneva, 1972)
9. C. Gencer, D. Gurpinar, Analytical network process in supplier selection: a case study in an electronic firm. *Appl. Math. Model.* **31**, 2475–2486 (2007)
10. C.L. Hwang, K. Yoon, *Multiple Attribute Decision Making Methods and Applications* (Springer, Berlin, 1981)
11. Y. Ju, A. Wang, T. You, *Emergency Alternative Evaluation and Selection Based on ANP, DEMATEL, and TL-TOPSIS* (Springer Science+Business Media, Dordrecht, 2014)
12. E.E. Karsak, S. Sozer, S.E. Alpteki, Product planning in quality function deployment using a combined analytic network process and goal programming approach. *Comput. Ind. Eng.* **44**, 171–190 (2002)
13. H.S. Kilic, S. Zaim, D. Delen, Selecting the best ERP system for SMEs using a combination of ANP and PROMETHEE methods. *Expert Syst. Appl.* **42**, 2343–2352 (2015)
14. C.L. Lin, W.W. Wu, A fuzzy extension of the DEMATEL method for group decision making. *Eur. J. Oper. Res.* **156**, 445–455 (2004)
15. C.-L. Lin, M.-S. Hsieh, G.-H. Tzeng, Evaluating vehicle telematics system by using a novel MCDM techniques with dependence and feedback. *Expert Syst. Appl.* **37**(10), 6723–6736 (2010)
16. C.T. Lin, C.B. Chen, Y.C. Ting, An ERP model for supplier selection in electronics industry. *Expert Syst. Appl.* **38**, 1760–1765 (2011)
17. A. Marasco, Third-party logistics: a literature review. *Int. J. Prod. Econ.* **113**(1), 127–147 (2008)
18. S. Onut, S.S. Kara, E. Isik, Long term supplier selection using a combined fuzzy MCDM approach: a case study for a telecommunication company. *Expert Syst. Appl.* **36**(2), 3887–3895 (2009)
19. C. Parkan, M.L. Wu, On the equivalence of operational performance measurement and multiple attribute decision making. *Int. J. Prod. Res.* **35**(11), 2963–2988 (1997)
20. T.L. Saaty, *The Analytic Hierarchy Process* (McGraw-Hill International, New York, 1980)
21. T.J. Saaty, *Decision Making in Complex Environments, The Analytical Hierarchy Process for Decision Making with Dependence and Dependence and Feedback* (RWS Publications, Pittsburgh, 1996)
22. J. Sarkis, A strategic decision framework for green supply chain management. *J. Clean. Prod.* **11**, 397–409 (2003)

23. J. Sarkis, S. Talluri, A model for strategic supplier selection, in *Proceedings of the 9th International IPSERA Conference*, ed. by M. Leenders (Richard Ivey Business School, London, 2000), pp. 652–661
24. J.-L. Shen, Y.-M. Liu, Y.-L. Tzeng, The cluster-weighted DEMATEL with ANP method for supplier selection in food industry. *J. Adv. Comput. Intell. Informat.* **16**(5), 256–266 (2012)
25. H.S. Shih, H.J. Syur, E.S. Lee, An extension of TOPSIS for group decision making. *Math. Comput. Model.* **45**, 801–813 (2007)
26. H.J. Shyur, H.S. Shih, A hybrid MCDM model for strategic supplier selection. *Math. Comput. Model.* **44**(8), 749–761 (2006)
27. S. Vinodh, S.S. Balagi, A. Patil, *A Hybrid MCDM Approach for Agile Concept Selection Using Fuzzy DEMATEL, Fuzzy ANP and Fuzzy TOPSIS* (Springer, London, 2015)
28. C.A. Weber, J.R. Current, W.C. Benton, Vendor selection criteria and methods. *Eur. J. Oper. Res.* **50**, 2–18 (1991)
29. W.W. Wu, Choosing knowledge management strategies by using a combined ANP and DEMATEL approach. *Expert Syst. Appl.* **35**(3), 828–35 (2008)
30. Z. Zhang, J. Lei, N. Cao, K. To, K. Ng, Evolution of supplier selection criteria and methods, in *The Second Globelics Conference Innovation Systems and Development: Emerging Opportunities and Challenges*, Beijing (2004)

Chapter 13

A Multi-Objective Optimization via Simulation Framework for Restructuring Traffic Networks Subject to Increases in Population

Enrique Gabriel Baquela and Ana Carolina Olivera

Abstract Traffic network design is a complex problem due to its nonlinear and stochastic nature. The Origin-Destiny Traffic Assignment Problem is particular case of this problem. In it, we are faced with an increase in the system vehicle population; and, we want to determine where to set the generating nodes and the traffic consumers, minimizing the current system and trying to reduce necessary investment. Performing optimizations in an analytical way in this kind of problems tends to be really complicated and a bit impractical, since it is difficult to estimate vehicle flows. In this chapter, we propose the use of a Multi-Objective Particle Swarm Optimization together with Traffic Simulations in order to generate restructuring alternatives that optimize both, traffic flow and cost associated to this restructure. This approach allows to obtain a very good approximation of the Pareto Frontier of the problem, with a fast convergence to the low infrastructure cost solutions and a total coverage of the frontier when the number of iterations is high.

Keywords Traffic net design • Particle swarm optimization • Metaheuristics • Traffic simulation • Simulated optimization

13.1 Introduction

Nowadays, population growth seems to have a direct impact on daily life due mainly to society progress. Traffic jams, pollution and parking problems are some of the most common problems in relation to traffic networks. In this chapter, we present a

E.G. Baquela (✉)

Facultad Regional San Nicolás, Universidad Tecnológica Nacional, San Nicolás de los Arroyos, Buenos Aires, Argentina

e-mail: ebaquela@frsn.utn.edu.ar

A.C. Olivera

Departamento de Ciencias Exactas y Naturales - Unidad Académica Caleta Olivia, Caleta Olivia, Santa Cruz, Argentina

Universidad Nacional de la Patagonia Austral. CONICET, Caleta Olivia, Santa Cruz, Argentina

e-mail: acolivera@conicet.gov.ar

methodology based on Optimization via Simulation (OvS) to select, from a group of nodes of a traffic network, the best alternative to increase the capacity of that network in the “Origin-Destiny Traffic Assignment Problem” (ODTAP), a subtype of the “Facility Location Problem” (FLP) and the “Network Traffic Design Problem” (NTDP) [17, 16]. Given a specific traffic network, which will suffer an increase in population, the main goal is to decide where to foment the installation of new urban infrastructure in order to minimize the cost of this and minimize the impact over the traffic system itself. In this context, we consider that the network already has a certain structure and a dynamic flow restricting possible configurations. To deal with the resolution of ODTAP, a Multi-Objective Particle Swarm Optimization Algorithm (MOPSO) was used together with Traffic Simulations (TS) in OvS based formalism.

The structure of this chapter goes as follow: in Sect. 13.1.1, a review of related works in the literature is presented; Sect. 13.2 explains the origin-destiny assignment problem and its formulation such as the optimisation problem; then, Sect. 13.3 shows an introduction to traffic simulation; Sect. 25.4 introduces the PSO algorithms for single and multi-objective problem; in Sect. 13.5, our optimisation approach is described; experiments and analysis of results are detailed in Sects. 16.6 and 16.7; finally, concluding remarks and future work are given in Sect. 16.8.

13.1.1 Literature Review

City growth reflects a progressive dynamic that is hardly ever planned, resulting in a decrease in the general performance of its sub-systems [41, 58]. In the case of the traffic sub-system, a population growth means more vehicles in the system, possible traffic jams, longer travel times, etc. [30, 41]. Even though this is not easy to control, it is possible to establish policies related to urbanization permits, to create industrial centers and to encourage activities in strategic zones, which will allow us to control the way the city grows and its associated traffic sub-system [18, 17, 28, 58]. In this line, the simulation [45, 40] reveals itself as a useful tool to quantify the impact of a certain traffic network topology and the corresponding population assignment and, also, other problems related to traffic systems [56, 9, 29, 53, 26].

In this work, we present the problem of deciding where to foment the installation of urban, industrial and/or commercial complexes so that it has a minimum effect over the total traffic system with the minimum possible cost in infrastructure. We take into account that the network already has a certain structure and a dynamic flow restricting possible configurations. We name this proposed decision problem restatement of the traffic network and the underlying problem Origin-Destiny Traffic Assignment Problem (ODTAP). The ODTAP is an underlying problem of the “Network Traffic Design Problem” (NTDP). The NTDP is the NP-Hard problem [16] of building a traffic network in such a way that it minimizes the performing function of the system, i.e. the mean travel time in general [57]. In this context, several literature works exist in relation to NTDP, some of them related to the ODTAP issue and urban planning.

The metaheuristic approach has proven to be useful for different optimization problems [43, 6, 33, 42, 47, 25, 54, 1, 32, 36, 59, 27]. For the NTDP, there are some works for which the solution is obtained by heuristics and bio-inspired algorithms [24, 7, 19, 20, 10, 13, 44].

In this chapter, we explore the Optimization via Simulation (OvS) capacity to obtain good solutions for ODTAP considering that this tool has proved to be useful to solve a lot of problems that, due to their complexity, are difficult to mold analytically.

In [26] the authors apply OvS optimizing with Particle Swarm Optimization (PSO) and simulating with the SUMO- Simulated of Urban Mobility tool [5] to solve the scheduling of traffic lights cycle. Kalganova et al. [34] uses genetic algorithms in an OvS to solve the traffic lights synchronization problem. Finally, Horvat and Tosic [31] utilize OvS together with genetic algorithms for the design of non-restrictive traffic networks in pre-existing networks.

Regarding what we have said, our work considers land use in the suburbs and cities in order to plan the urban expansion and predicts its growth considering its final impact on the traffic organization. The proposed methodology combines a popular traffic simulator SUMO [5] with “Speed-constrained Multi-objective Particle Swarm Optimization” (SMPSO) in a Optimization via Simulation context [22] in order to improve the traffic network. The propose methodology take advantage of the infrastructure in the urban sprawl. Our objectives are minimizes the impact on the landscape to obtain accessibility and mobility facilities. The experiments and comparisons with other techniques reveal that our proposed OvS approach obtains significant profits in terms of traffic network improvement. By its own nature, the OVS algorithm presented here can be easily adapted to solve different multi-objective problems [23, 21].

13.2 Origin-Destiny Traffic Assignment Problem

In this section, we define our case of study (traffic systems) and the ODTAP. Then, we expose the characteristics of the problem to be optimized.

13.2.1 Traffic Systems

A traffic system is a complex system made up of at least two elements: a traffic network and a traffic demand.

A traffic network is an oriented graph in which the edges represent the streets, and the nodes, the changing points of those streets. In Fig. 13.1, we can see a simple traffic network, composed of two one-way streets at each intersected at the corner. It is represented by a graph with 4 edges and 5 nodes, in which nodes “O” are traffic origins, nodes “D” are traffic destinations and nodes “I” are corners. As it can be

observed in the previous figure, there are different types of nodes and edges in one traffic network. Each edge is anathematized by a group of parameters: its length, the number of tracks of the traffic route, maximum circulating speed, and so on. The nodes are also defined by certain features which determine their participation in the model: kind of node (origin, destiny, lane extension, corner, etc.) and spatial coordinates, and so on. In our case, we are interested in identifying origin nodes or demand destiny nodes, as they generate or receive traffic flow. This kind of nodes needs to have associated information about maximum capacity and its rate of emitting and receiving vehicles.

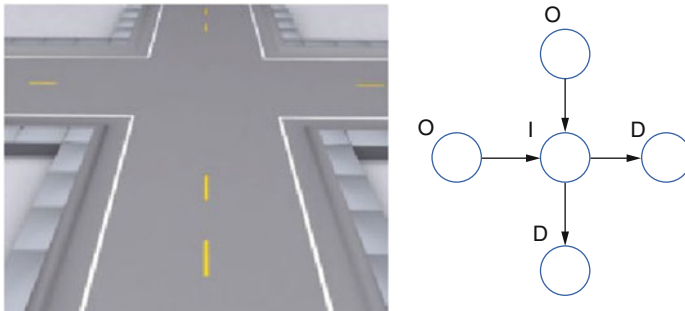


Fig. 13.1 Representation of streets intersection

Traffic demand, on the other hand, is the set of all circulating vehicles in the system (taking into account the routes they follow and their starting time). In a practical way, traffic demand is more difficult to deal with than the network structure. As it is, in a real implementation it is highly difficult not to say impossible to know the routes and starting points of the total number of vehicles of the system. Besides, generally, traffic studies normally imply having to determine how routes configuration can vary against changes in the system, turning the previous group useless. That is the reason why the demand tends to be characterized by a generating group, like traffic flows (which is the characterization used in this work). A traffic flow is a group of vehicles that in a specific time window circulating from a node i to a node j . The previous definition does not include the routes followed by the vehicles; so, flows tend to be invariable to most of the changes to be done in a traffic system. Changes which imply an alteration in the flows (as it is the case of ODTAP) turn to be easier to handle, because one only has to vary the number of vehicles associated to each flow. In order to evaluate a traffic system, it is necessary to generate the routes for the involved vehicles taking into account the flows, for which you have several mechanisms: proportional assignment, shorter paths, balancing assignments.

13.2.2 Origin-Destiny Traffic Assignment Problem

“Origin-Destiny Traffic Assignment Problem” (ODTAP) [4] tries to absorb the impact of a variation in the traffic demand (by increasing or redistributing flows) through a restructuration of the origin nodes and destinations of traffic flows. For instance, if you want to determine where to set a new shopping mall in the city. In this situation, it will not necessarily increase the number of vehicles, but it will modify the routes followed by the same vehicles at some particular times. The decision of where to set the new mall (regulated by the city hall through permits and authorizations) can be assimilated to vehicle reassignment to new traffic flows that have as one of their extremes the shopping mall (either the destiny or the origin, depending on the time frame). Another example can be the installation of an industrial depot (in this the city hall directly controls the location). Apart from redirecting vehicle routes already existing in the system, the installation of the industrial depot will probably generate new vehicle flows from an to other cities (increased by the total number of vehicles circulating). Once again, from a traffic point of view, this is translated in a restructuration of existing traffic flows and in the generation of new ones having the depot as origin and destiny. As a final example we can name the setting off a new neighborhood in town, which will have a similar effect as the previous of producing new traffic flows arriving and departing from and to the neighbor.

The three examples stated in the previous paragraph become new traffic origins and destinies. In general, if the change in the system is carried out in an organized way, it is possible to establish areas in which this new origins and destinies can establish themselves, either through regulations or plans encouraging its reside). This tend to have a meaningfully high infrastructure cost, since they demand zone paving, basic service supply, an improvement in the entrance, and so on. Therefore, the problem is now how to do this assignment (i.e. selecting the nodes of a traffic network which will function as the new origins and destinies) with the minor possible effect on current traffic, on the one hand, and using the less possible amount of money for infrastructure. Measuring the cost impact is quite simple, but evaluating the impact on the traffic system is much more complicated. It normally uses a mean time metric, thus, the size of the impact on traffic is estimated by time variations of mean time of all the system vehicles.

Formally, the ODTAP is defined on n nodes belonging to the set $N = \{1, \dots, n\}$, with arc (edges) set E , and weight costs w_k with $k = 1, \dots, m$; associated with the arcs, then, the ODTAP can be resumed as follow:

$$\min Z_1 = \sum (t_{ijp} \cdot (Pop_{ijp} + X_{ijp})) \quad (13.1)$$

$$\min Z_2 = \sum (c_i \cdot X_i) \quad (13.2)$$

subject to

$$\sum (X_i + Pop_i) \leq \text{Max Population at Origin } i \quad (13.3)$$

$$\sum (X_j + Pop_j) \leq \text{Max Population at Destination } j \quad (13.4)$$

$$\sum (X_{ijp}) = \text{Increase in Population} \quad (13.5)$$

Where, $i \in N$ are the origins, $j \in N$ are the destinies, p are the paths that connect i with j , Pop_{ijp} is the current population that needs to travel from i to j through the path p , X_{ijp} is the new assignment of population that will need to travel from i to j through the path p , t_{ijp} is the travel time from i to j in the path p of the population and c_i infrastructure cost necessary to support a population growth X_i in node i . It is important to note that this characterization of the ODTAP should be enough for small instances in simple models. However, there is not always accurate information available about traffic travel times associated with the land use in urbanization planning. The genuine function of traffic travel time depends on many uncontrollable factors such as the shape of the roads, the amount of the street, the velocity of the vehicles, etc. Moreover, traffic congestion can be an important factor in the total journey of the population. For these reasons, the total travel time can be reformulated in Eq. (13.6).

$$\min Z_1 = \sum (t_{ijp}((Pop_{i1} + X_{i1}), \dots, (Pop_{in} + X_{in})) \cdot (Pop_{ijp} + X_{ijp})) \quad (13.6)$$

Hence, it is considered the impact between the origin-destiny assignments into the traffic network and how it affects the Land Use planning

13.3 Traffic Simulations

Because we want to represent correctly the effect of variations in vehicle flows have on mean travel time, we will test that assignation by doing a simulation of the traffic flow. There are several ways in which you can simulate traffic flow in a traffic network, but in this work we choose to use continuum microscopic simulations.

A microscopic traffic simulation is a type of simulation of discrete time in which each vehicle behavior is modeled and simulated individually. Each vehicle in the traffic system is characterized with an identification code (“id”), a group of constant parameters regulating the route to follow, the moment in which the trip starts, maximum speed and the rest of the vehicle circulation criteria, and a group of variables with vehicle and speed information (they are updated in every interval t). The behavior of each vehicle follows simple laws, represented in an equation of the type stimuli-response where the main factor that determines the behavior of vehicles is the vehicle situated in front. These kinds of logics are known as “Car-Following” [40]. An example of this logic for a vehicle i is the following:

- If there is no car in front (from the vehicle position of vehicle i to a critical distance), accelerate until reaching maximum speed.

- If there is one car in front and its speed is lower than that of vehicle i , slow down to equal the speed of that vehicle.
- If there is a car front and its speed is higher or the same as that of vehicle i , accelerate to reach the same speed or to reach the maximum speed of vehicle i , whatever is lower.

Apart from interactions with other vehicles, interactions with the rest of the system elements are added, like the traffic lights system.

Taking into account each vehicle information, it is possible to obtain afterwards variables aggregated to the system, as average speed, maximum waiting time [11, 37].

The main advantage this type of simulations offers is that modeling all its elements is comparatively easier than in the rest of the simulations, without the need of making assumptions about global system behavior. In fact, global behavior emerges as a consequence of interactions between each of its components.

13.4 Multiobjective Particle Swarm Optimization

As it was previously mentioned, with the objective of optimizing the ODTAP we use a multi-objective version of the “Particle Swarm Optimization” (PSO). In this section, we make an introduction to this metaheuristic and its adaptation to solve multi-objective problems.

13.4.1 Particle Swarm Optimization

The PSO algorithm first appeared in 1995 [35] and since then it has been used in a high number of problems [15, 2, 52, 3, 51]. PSO is based on bird flock behavior when they go searching for their food. In a flock, you can observe individual and group behavior. PSO algorithm is a population optimization algorithm where in each iteration we have a group of potential solutions (particles expressed in PSO terminology) that function as the birds of the flock. Every particle is defined based on their position p and its speed v in a space $n - dimensional$. Interaction after interaction, each one of them evolve according to the historic individual information recollected and to global historic information of the whole flock, updating its position and speed. This evolution is regulated by Eqs. (13.7) and (13.8):

$$v_{i,j}(t+1) = w \cdot v_{i,j}(t) + c_1 \cdot r_1 \cdot (p_{i,j}(t) - x_{i,j}(t)) + c_2 \cdot r_2 \cdot (p_{g,j}(t) - x_{i,j}(t)) \quad (13.7)$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \quad (13.8)$$

Where w is the inertia factor (which regulates the updating parameters speed), $v_{i,j}$ is the component of particle i in dimension j , c_1 and c_2 are the importance of individual and global information in each update, r_1 y $r_2 \sim U(0,1)$ represent the differential acceleration of each particle, p_i is the best value found by particle i ($pbest$) and p_g is the best solution found in the neighborhood ($gbest$). During each interaction, the speed value is updated first, and then the position value of each particle.

Even though it is a kind of evolutionary algorithm, PSO presents the distinctive feature that the individuals belonging to the population do not compete with each other to monopolize the following interactions, but they even cooperate to find the global optimum.

13.4.2 Multi-Objective Particle Swarm Optimization

The original definition of PSO was thought to deal with mono-objective problems. However, there are several adaptations to deal with problems in which it is required to optimize several objectives simultaneously [50, 48, 38, 46].

In a multi-objective problem, the focus is no longer finding the global optimum but to obtain the best approximation to the Pareto Frontier of the system. For each point in the “Decision Space” there is an associated point in the “Objective Space”, being the latter a m -dimensional space with m equal to the number of objectives of the problem. Given that it exists several “Trade-Off” possible among the different objectives, the function of the optimization algorithm is finding the section in the frontier of the “Objective Space” where these “Trade-Off” are better than any other belonging to that space. In multi-objective optimization terminology, a solution i dominates a solution j if it is at least better in one of their objectives and better or the same in the rest. The aim of a multi-objective optimization algorithm is, then, to find the group of all solutions that are not dominated by any other solution of the space (thus, they are better than the rest) [38].

In this work, we decided to use the “Speed-constrained Multi-objective PSO” (SMPSO). This algorithm was presented in [14] and it is an adaptation of “Optimized Multi-Objective PSO” (OMOPSO) presented in [49]. Both have proven to have a good performance in comparison to other versions of the multi-objective PSO, and even compared to NSGA-II [14, 8]. The main problem in any multi-objective alteration based on Pareto Frontiers of the PSO consists in how to select the leader of each interaction ($gbest$). Both algorithms handle the leader selection with a fix size list of leader solutions, selected from the group of non-dominated solutions. In each interaction, solutions are added or eliminated from the list following the non-dominating criteria. To all the leaders, a “Crowding Factor” is calculated. In case the size of the list increases, the solutions with the worst “Crowding Factor” value are eliminated; this value performing as second classificatory factor of solutions. To estimate the speed v_i of each swarm particle, a leader is chosen through a binary tournament based on the crowding value of the leaders.

Apart from this leader selection mechanism, the OMOPSO and the SMPSO have other features. The calculation of non-dominance follows non-strict dominance criteria o ε -dominance. is considered to be dominant with respect to a solution s if for all objectives $f(q)/(1 + \varepsilon) \leq f(s)$ and to at least one objective $f(q)/(1 + \varepsilon) < f(s)$. Also, to avoid bias, the parameters c_1 and c_2 are selected randomly. Finally, mutation process of solutions is incorporated, taken from the namesake process used in Genetic Algorithms.

Unlike OMOPSO, SMPSO also adds a speed calculation restriction; with the goal of avoiding particles positioning varies from its feasible extremes without evaluating intermediate points. To do this, speeds are restricted according to the Eqs. (13.9) and (13.10).

$$-delta_j \leq v_{i,j}(t) \leq delta_j \quad (13.9)$$

$$delta_j = \frac{upperLimit_j - lowerLimit_j}{2} \quad (13.10)$$

13.5 Optimization Framework

In this section we explain the OvS Procedure developed to solve the Multi-Objective version of ODTAP.

13.5.1 General Procedure

In order to solve the ODTAP, we have chosen a formalism based on OvS [23]. In the OvS, the evaluation of the goodness of solutions is done by the execution of a simulator, which functions as the generator of necessary values to calculate the objective function. Both the optimizer and the simulator are independent from each other, connecting themselves in a black box way, having a system functioning as a controller between the two (Fig. 13.2). Apart from regulating the process, the controller has the function of translating the resulting solutions brought by the optimizers to configuration parameters for the simulator, at the same time it converts this last one outputs (data samples) into a value the optimizer can accept as an aptitude or fitness function. In this context, SUMO is used for simulation purpose [5] and the SMPSO to the optimization part. This framework is adapted from a framework used to solve the mono-objective version of the ODTAP, presented in [4], in which it replaces the use of Genetic Algorithms by the PSO. The OvS procedure was implemented in the popular software R [39, 55] with the library RSUMO (<http://www.modelizandosistemas.com.ar/p/rsumo>).

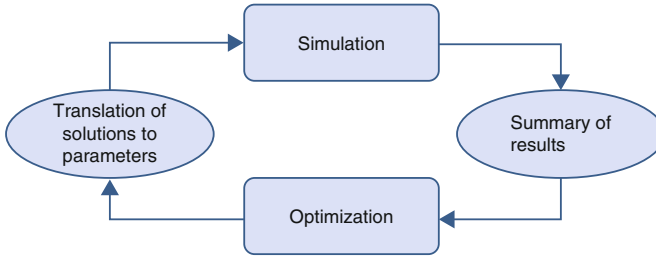


Fig. 13.2 OvS procedure as black box

Figure 13.3 shows the interrelations between the R language, SUMO package and the PSO Procedure. Defined current traffic net and current traffic density, in order to absorb population growth and restrictions on the capacity increase in the nodes, the procedure generates SMPSO solutions to assess, in the form of increases in capacity per node. These solutions are translated into traffic flows and used to run the simulation. When the simulation process ends, the travel time statistics are consolidated in an average value of travel time and reentered the SMPSO to determine the quality of each solution and generate new solutions in the next iteration.

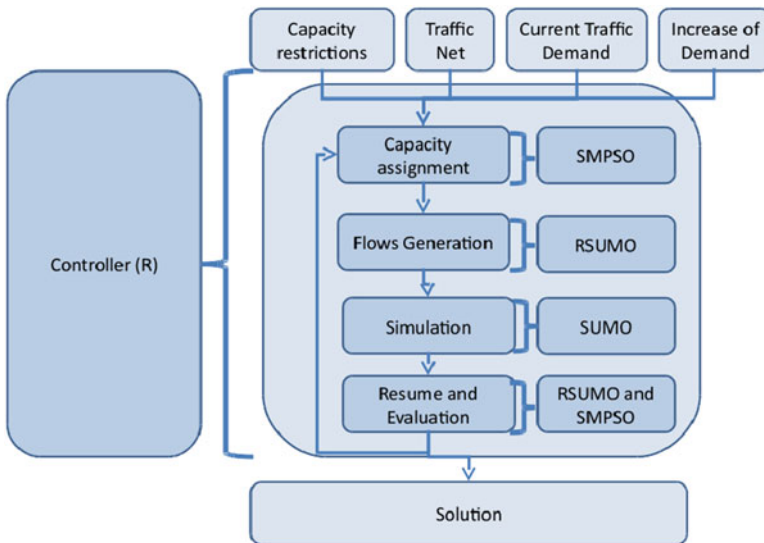


Fig. 13.3 OvS procedure

13.5.2 Solution Evaluation

In our model, SMPSO algorithm only assigns values to variables $X_{i,j}$. That is to say, it defines the population growth (or decrease) in each origin-destiny pair of the

potential origin-destiny pair group in which to invest on infrastructure. Based on this assignation, we can calculate each traffic flow $Pop_{i,j} + X_{i,j}$ (keeping $Pop_{i,j}$ constante durante todas las evaluaciones). constant during all evaluation). With the group of traffic flows, SUMO can estimate the distribution of that flow among every path p that link i with j , and then assign starting moments to each of the vehicles involved in the routes. Once each vehicle is programmed, simulations starts and statistics are recollected about mean travel time.

The evaluation of the objective cost is much simpler. You only have to compute the number of vehicles assigned to each node and multiply it with the infrastructure cost. In case it is requested, you can modify the calculation by assuming the non-linearity of these costs, making them vary step by step with respect to the number of assigned vehicles, without altering the optimization model.

13.6 Experiments

In this section, we detail the experiments carried out to evaluate the quality of the solutions of the proposed optimization algorithm.

13.6.1 Scenarios

Building scenarios can be divided in two stages: construction of traffic networks and generation of initial traffic flows. To test the quality of the algorithm, a group of ten scenarios was designed based on traffic network like the “grid” type. Its network presents the peculiarity of having a great number of possible routes between two nodes, whichever they are. Doing small variations in this network, you can get traffic networks similar to those of various urbanization types. Summarizing, the transformation operations carried out were:

1. Given a grid network of size $m \cdot n$, we determined a number of changes equals to $\sqrt{\min(m,n)}$.
2. For each change to be done, we set their type: “edge change” or “node change”.
 If the kind of change is “edge change”, we select a random edge from the network and we eliminate it.
 If the king of change is a “node change”, we select a random node from the network and we eliminate it and all the edges attached to it.

An example of a network construction can be seen in Fig. 13.4.

On the other hand, once it is defined the traffic network, it is necessary to generate the traffic demand associated to the situation previous to the population increase to be dealt with (every Pop_{ijp}). To do that, starting from the traffic networks previously created, an initial number of vehicles circulating during an hour of a simulated time in a random value between $100 \cdot \max(m,n)$ and $10 \cdot m \cdot n$ was fixed. Having the total

number of vehicles, the origin and destiny of each one are randomly raffled from the network nodes through a non-repository sampling

Regarding the maximum capacity and cost of each node, the first one was fixes in a value similar to its current population assignation increased by a random factor contained between 1.01 and 2.00. The unitary costs of increased capacity were randomly fixed too, taking some range value [10 : 100].

In total, we worked with 3 initial traffic networks, sized $10 \cdot 10$, $25 \cdot 50$ and $50 \cdot 75$, from which we generated 3 other daughter networks, according to the previously described technique. Afterwards, for each of these 9 networks, 3 random flows were created, leaving a total of 27 traffic scenarios. To every scenario a maximum capacity of every node and increase costs were randomly assigned. Every traffic scenario was evaluated with a population growth of 1.05 and 1.15, leaving 54 cases of study defined in total.

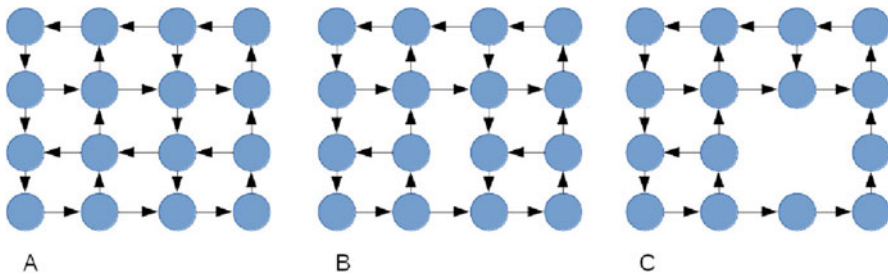


Fig. 13.4 (a) Grid net. (b) Elimination of one arc. (c) Elimination of one node

13.6.2 Tests

To evaluate how well our algorithm approximates to the problem Pareto frontier, for each scenario 10 runs were carried out. To create the Pareto frontier associated to the scenario, all generated solutions by each 10 run were consolidates and the Pareto frontier of this new data was calculated. This gave us a Global Estimated Pareto frontier for the total scenario, made up by the non-dominance solutions found in every run. That global estimated Pareto frontier is the one used as a pattern to measure the quality of the found solutions en each individual run.

Two solution quality indicators were measured: “Convergence” and “Uniformity” “Convergence” is calculated by averaging the minimum Euclidean distance of each point of the Pareto frontier obtained in one run of the algorithm with the Global Estimated Pareto frontier. This method of estimating the convergence is called “Generational Distance”. The complete index is shown in Eq. (13.11):

$$\text{generational distance} = \frac{\sqrt{\sum_{i=1}^N d_i^2}}{N} \quad (13.11)$$

Where N is the number of solutions to be compared with the Pareto Frontier and d_i is the minimum distance from solution i to the Pareto Frontier.

Moreover, the “uniformity” is calculated by Eq. (13.12).

$$\text{uniformity} = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + \bar{d}(N-1)} \quad (13.12)$$

Where d_f and d_l are the minimum Euclidean distances between the ends of the Global Estimated Pareto Frontier and the estimated Pareto Frontier in current run, d_i is the distance between the solution i and the solution $i + 1$ from the estimated Pareto Frontier in current run, \bar{d} is the average of $N - 1$ distances d_i and N is the number of points in the estimated Pareto Frontier in current run.

For a further comparison, we have studied the performance of the NSGA-II [12] for the same experimental procedure as our proposal.

13.6.3 Algorithm Parameters

The parameters under which the SMPSO was executed are the following:

- *Probability of Mutation* = 0.5
- $w = 0.4$
- $\varepsilon = 0.05$
- *Swarm Size* = 50
- *Max Number of Generations* = 100

On the other hand, the simulation time to evaluate each solution was set in 1.200 s.

13.7 Results

In this section, we present the results of the experiments done.

13.7.1 Convergence

The average of generational distance between found solutions by the algorithm and the global estimated Pareto frontier is kept in low values, with a deviation coefficient

oscillating between 0.27 and 0.41. In Table 13.1 it can be observe convergence values for one of the scenarios of size $10 \cdot 10$ with an increase in vehicle population of 1.15. The convergence average of this group is of 0.40384 and the desviation coefficient is 0.38759. It can be noticed that the majority of the runs were kept in the same generational distance range, except two atypical observations in one of which we can see the effect of the presence of local optimum (item 4). The lower generational distance in item 3 is because a group of solutions kept a low mean travel time with a low infrastructure cost. For this specific scenario, the algorithm was run once more with a swarm size of 250 individuals. The convergence results are shown in Table 13.2. An improvement in the indicators can be noticed.

Table 13.1 Convergence and Uniformity of grid $10 \cdot 10$ and population increase of 1.15

Run	Convergence	Uniformity
1	0.3533009	0.2291444
2	0.4108350	0.4742433
3	0.0837602	0.4537517
4	0.7143926	0.3012058
5	0.3307537	0.1526166
6	0.4584290	0.1768969
7	0.3969564	0.2108491
8	0.4987557	0.3343977
9	0.4229219	0.2606873
10	0.3682840	0.1414944

Table 13.2 Convergence and Uniformity of grid $10 \cdot 10$ and population increase of 1.15 (swarm size = 250)

Run	Convergence	Uniformity
1	0.2106560	0.1374553
2	0.3108199	0.3622892
3	0.1789973	0.2721308
4	0.2022944	0.2191117
5	0.2454660	0.1283510
6	0.3290329	0.3134578
7	0.4945728	0.2041503
8	0.0927502	0.3940818
9	0.2933584	0.1594452
10	0.0771648	0.3538501

13.7.2 Uniformity

Uniformity finds its average in low values, but with a higher desviation coefficient than in convergence (oscillates between 0.33 and 0.46). Table 13.1 shows uniformity values for the presented case in Sect. 13.7.1. It can be observed that the behavior here is different, with a significant number of solutions with good uniformity values, and some solutions with a high uniformity. Once it was run again the algorithm with a swarm size of 250, it was impossible to observe significant improvements in this indicator.

13.7.3 Evolution of the Solutions Generated by the Algorithm

In Figs. 13.5 and 13.6 you can see the solutions from iterations 25 and 100 for one run of the algorithm, compared with the Pareto Frontier. The adjustment in the iteration 100 is very good, with almost all solutions belonging to the Pareto Frontier. It further notes that while there is not a great diversity of solutions, the Pareto frontier is swept almost entirely. At iteration 25 can see that while convergence of the solutions is very low, the algorithm found several individuals who belong to the frontier. Moreover, in Figs. 13.7 and 13.8 you can see the evolution of the population and the Estimated Pareto Frontier in iterations 25, 50, 75 and 100. In the earlier iterations, the dispersion of solutions is big and there are few solution in the estimated Pareto Frontier. In the last iterations, the dispersion is lesser and the majority of solutions are near to the estimated Pareto Frontier.

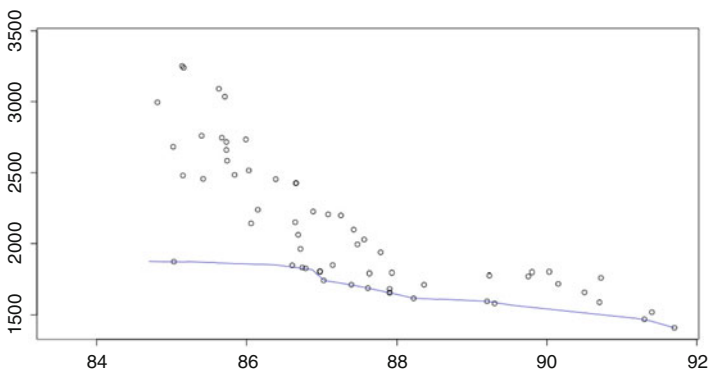


Fig. 13.5 Iteration 25. The line is the Pareto Frontier, points are the individual of the iteration

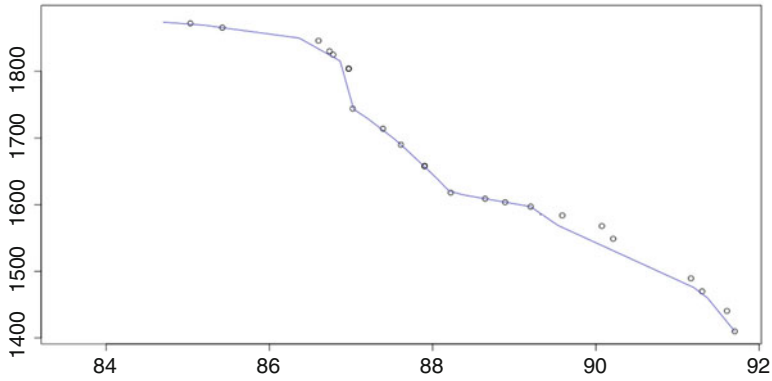


Fig. 13.6 Iteration 100. The line is the Pareto Frontier, points are the individual of the iteration

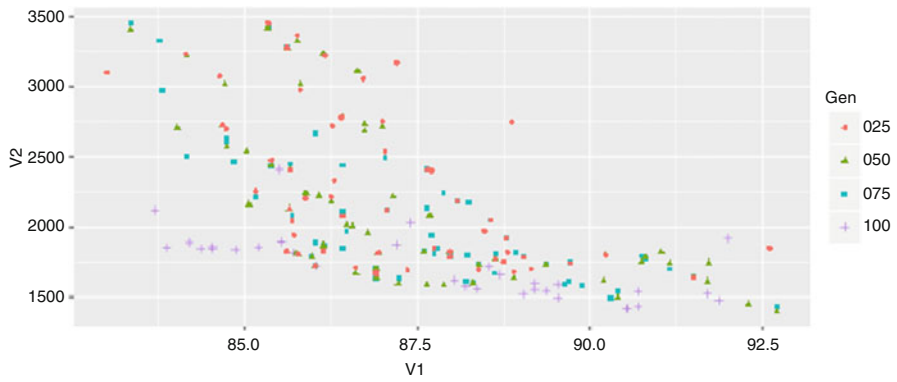


Fig. 13.7 Evolution of population

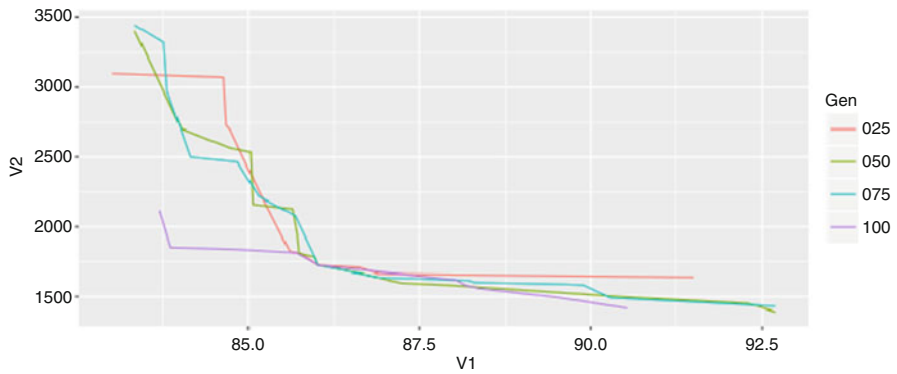


Fig. 13.8 Evolution of estimated Pareto Frontier

It's interesting see that the convergence to the Pareto Frontier is faster for the combination of “low cost-high travel time” than the “high cost-low travel time”. This is a consequence of the structure of the problem: the infrastructure costs are constant, but the mean speed is a dynamic function of the vehicles assignment in the origins and destination of traffic flows.

13.7.4 Comparison with NSGA-II Metaheuristic

The same set of experiments were running under a NSGA-II metaheuristics, in order to check the quality of SMPSO solution. In Table 13.3 are shown the result of 10 runs for the same scenario that is shown in Table 13.2.

Table 13.3 Convergence and Uniformity of grid 10 · 10 and population increase of 1.15 (swarm size = 250)

Run	Convergence	Uniformity
1	0.2451904	0.2159275
2	0.4593215	0.3635322
3	0.4765746	0.1010504
4	0.2518910	0.3037142
5	0.2027211	0.3484107
6	0.4149186	0.1499699
7	0.1445920	0.1737689
8	0.1791007	0.2756582
9	0.3145456	0.4711506
10	0.3351634	0.3871142

According to the “Wilcoxon Signed Rank” the convergence of solution founded by the SMPSO algorithm are better than the NSGA-II, but for a low margin (p -value $\cong 0.03$). The uniformity is similar for both algorithms.

13.8 Conclusions

In this chapter, we presented an algorithm that combines SMPSO with a traffic simulation to solve a multi-objective version of the ODTAP. The obtained results show a good performance, being able to find a great approximation to the problem Pareto frontier with a swarm size not so big. The results get better when the swarm size increases, scarifying an increase in the computation time. The SMPSO also showed a slightly better performance than the NSGA-II for this problem.

Using simulation as an evaluating function of the optimization algorithm lets us faithfully represent the effect of policies to be implemented in the system, without the distorting effects of an analytical simplification. The algorithm has showed a fast convergence rate for the “low cost-high travel time” side of the Pareto Frontier.

This is consequence of the dynamic nature of the speed of cars in the system, which varies in function of the assignments in the origins and destinations. The use of simulations allows us to evaluate this dynamic in a easy way.

Acknowledgements The work of Baquela E. G. was supported by the Universidad Tecnologica Nacional, under PID TVUTNSN0003605. Olivera A. C. thanks to ANPCyT for grant PICT 2014-0430, CONICET (PCB-I), and Universidad Nacional de la Patagonia Austral for PI 29/B168.

References

1. S.K. Azad, O. Hasancebi, S.K. Azad, Upper bound strategy for metaheuristic based design optimization of steel frames. *Adv. Eng. Soft.* **57**, 19–32 (2013)
2. A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part I: background and development. *Nat. Comput.* **6**(4), 467–484 (2007)
3. A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Nat. Comput.* **7**(1), 109–124 (2008)
4. E.G. Baquela, A.C. Olivera, Combining genetic algorithms with traffic simulations for restructuring traffic networks subject to increases in population, in *International Conference on Metaheuristics and Nature Inspired Computing 2014* (2014)
5. M. Behrisch, L. Bieker, J. Erdmann, D. Krajzewicz, Sumo – simulation of urban mobility: an overview, in *SIMUL 2011, The Third International Conference on Advances in System Simulation*, Barcelona, 2011, pp. 63–68
6. J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes* (2011). <https://cs.gmu.edu/~sean/book/metaheuristics/>
7. L. Caggiani, M. Ottomanelli, Traffic equilibrium network design problem under uncertain constraints. *Procedia Soc. Behav. Sci.* **20**, 372–380 (2011)
8. P. Carrasqueira, M.J. Alves, C.H. Antunes, A bi-level multiobjective pso algorithm, in *Evolutionary Multi-Criterion Optimization – 8th International Conference*, Apr 2015
9. H.W. Casey, Simulation optimization of traffic light signal timings via perturbation analysis, Ph.D. thesis, Faculty of the Graduate School of the University of Maryland, College Park, 2006
10. H. Ceylan, M. Bell, Genetic algorithm solution for the stochastic equilibrium transportation networks under congestion. *Transp. Res. B* **39**, 169–185 (2005)
11. D. Chowdhury, L. Santen, A. Schadschneider, Statistical physics of vehicular traffic and some related systems. *Physics Report* **329** (2000), pp. 199–329
12. K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, in *Parallel Problem Solving from Nature PPSN VI. Lecture Notes in Computer Science*, vol. 1917 (Springer, Berlin, 2000), pp. 849–858
13. S. Dinu, G. Bordea, A new genetic approach for transport network design and optimization. *Bull. Pol. Acad. Sci. Tech. Sci.* **59**(3), 263–272 (2011)
14. J.J. Durillo, J. Garcia-Nieto, A.J. Nebro, C.A.C. Coello, F. Luna, E. Alba, Multi-objective particle swarm optimizers: an experimental comparison, in *Evolutionary Multi-Criterion Optimization 2009* (2009)
15. R.C. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, in *Proceedings of the 2001 Congress on Evolutionary Computation, 2001*, vol. 1 (2001), pp. 81–86
16. R.Z. Farahani, E. Miandoabchi, W. Szeto, H. Rashidi, A review of urban transportation network design problems. *Eur. J. Oper. Res.* **229**, 281–302 (2013)

17. J.A. Ferreira, B. Condessa, Defining expansion areas in small urban settlements an application to the municipality of Tomar (Portugal). *Landsc. Urban Plan.* **107**, 281–302 (2012)
18. J.A. Ferreira, B. Condessa, J.C. e Almeida, P. Pinto, Urban settlements delimitation in low-density areas an application to the municipality of Tomar (Portugal). *Landsc. Urban Plan.* **97**(3), 156–167 (2010)
19. H. Fredrik, Towards the solution of large-scale and stochastic traffic network design problems. Master's thesis, Uppsala Universitet, 2010
20. T.L. Friesz, H.-J. Cho, N.J. Mehta, R.L. Tobin, G. Anandalingam, A simulated annealing approach to the network design problem with variational inequality constraints. *Transp. Sci.* **26**(1), 18–26 (1992)
21. M. Frutos, A.C. Olivera, F. Tohmé, A memetic algorithm based on a NSGAI scheme for the flexible job-shop scheduling problem. *Ann. Oper. Res.* **181**, 745–765 (2010)
22. M.C. Fu, Optimization via simulation: a review. *Ann. Oper. Res.* **53**, 199–247 (1994)
23. M.C. Fu, Optimization for simulation: theory vs. practice. *INFORMS J. Comput.* **14**(3), 192–215 (2002)
24. M. Gallo, L. D'Acerno, B. Montella, A meta-heuristic algorithm for solving the road network design problem in regional contexts. *Procedia Soc. Behav. Sci.* **54**, 84–95 (2012). Proceedings of EWGT2012 – 15th Meeting of the EURO Working Group on Transportation, September 2012, Paris
25. A.H. Gandomi, X.-S. Yang, S. Talatahari, A.H. Alavi, 1 – metaheuristic algorithms in modeling and optimization, in *Metaheuristic Applications in Structures and Infrastructures*, ed. by A.H. Gandomi, X.-S. Yang, S. Talatahari, A.H. Alavi (Elsevier, Oxford, 2013), pp. 1–24
26. J. Garcia-Nieto, A. Olivera, E. Alba, Optimal cycle program of traffic lights with particle swarm optimization. *IEEE Trans. Evol. Comput.* **17**, 823–839 (2013)
27. H.C. Gomes, F. de Assis das Neves, M.J.F. Souza, Multi-objective metaheuristic algorithms for the resource-constrained project scheduling problem with precedence relations. *Comput. Oper. Res.* **44**, 92–104 (2014)
28. N. Haregeweyn, G. Fikadu, A. Tsunekawa, M. Tsubo, D.T. Meshesha, The dynamics of urban expansion and its impacts on land use land cover change and small-scale farmers living near the urban fringe: a case study of Bahir Dar, Ethiopia. *Landsc. Urban Plan.* **106**(2), 149–157 (2012)
29. C. He, N. Okada, Q. Zhang, P. Shi, J. Li, Modelling dynamic urban expansion processes incorporating a potential model with cellular automata. *Landsc. Urban Plan.* **86**(1), 79–91 (2008)
30. C. He, J. Tian, P. Shi, D. Hu, Simulation of the spatial stress due to urban expansion on the wetlands in Beijing, China using a GIS-based assessment model. *Landsc. Urban Plan.* **101**(3), 269–277 (2011)
31. A. Horvat, A. Tomic, Optimization of traffic networks by using genetic algorithms. *Elektrotehnicki Vestnik* **79**, 197–200 (2012)
32. M.K. Jha, M. Head, S.P. Gar, 23 – metaheuristic applications in bridge infrastructure maintenance scheduling considering stochastic aspects of deterioration, in *Metaheuristic Applications in Structures and Infrastructures*, ed. by A.H. Gandomi, X.-S. Yang, S. Talatahari, A.H. Alavi (Elsevier, Oxford, 2013), pp. 539–556
33. J. Jin, T.G. Crainic, A. Lokketangen, A cooperative parallel metaheuristic for the capacitated vehicle routing problem. *Comput. Oper. Res.* **44**, 33–41 (2014)
34. T. Kalganova, G. Russell, A. Cumming, Multiple traffic signal control using a genetic algorithm, in *Artificial Neural Nets and Genetic Algorithms* (Springer, Vienna, 1999), pp. 220–228
35. J. Kennedy, R. Eberhart, Particle swarm optimization, in *International Conference on Neural Networks* (IEEE Service Center, Piscataway, NJ, 1995), pp. 1942–1948
36. J. Kratica, An electromagnetism-like metaheuristic for the uncapacitated multiple allocation p-hub median problem. *Comput. Ind. Eng.* **66**(4), 1015–1024 (2013)
37. S. Krauss, Microscopic modeling of traffic flow: investigation of collision free vehicle dynamics. Hauptabteilung Mobilität und Systemtechnik des DLR Köln, 1998

38. V. Kumar, S. Minz, Multi-objective particle swarm optimization: an introduction. *Smart Comput. Rev.* **4**, 335–353 (2014)
39. D.T. Lang, *XML: Tools for parsing and generating XML within R and S-Plus*, R package version 3.95-0.1 (2012)
40. Y. Li, D. Sun, Microscopic car-following model for the traffic flow: the state of the art. *J. Control Theory Appl.* **10**(2), 133–143 (2012)
41. Y. Li, X. Zhu, X. Sun, F. Wang, Landscape effects of environmental impact on bay-area wetlands under rapid urban expansion and development policy: a case study of Lianyungang, China. *Landscape Urban Plan.* **94**(3,4), 218–227 (2010)
42. A. Liefvooghe, S. Verel, J.-K. Hao, A hybrid metaheuristic for multiobjective unconstrained binary quadratic programming. *Appl. Soft Comput.* **16**, 10–19 (2014)
43. S. Luke, *Essentials of Metaheuristics*, 2nd edn. (Lulu, 2013). <https://cs.gmu.edu/~sean/book/metaheuristics/>
44. E. Miandoabchi, R.Z. Farahani, Optimizing reserve capacity of urban road networks in a discrete network design problem. *Adv. Eng. Softw.* **42**(12), 1041–1050 (2011)
45. K. Nagel, M. Schreckenberg, A cellular automaton model for freeway traffic. *J. Phys. I* **2**, 2221–2229 (1992)
46. N. Nedjah, L. de Macedo Mourelle, Evolutionary multi objective optimisation: a survey. *Int. J. Bio-Inspired Comput.* **7**(1), 1–25 (2015)
47. E. Olivares-Benitez, R.Z. Rios-Mercado, J.L. Gonzalez-Velarde, A metaheuristic algorithm to solve the selection of transportation channels in supply chain design. *Int. J. Prod. Econ.* **145**(1), 161–172 (2013)
48. M.J. Reddy, D.N. Kumar, Multi-objective particle swarm optimization for generating optimal trade-offs in reservoir operation. *Hydrol. Process.* **21**, 2897–2909 (2007)
49. M. Reyes-Sierra, C.A.C. Coello, Improving pso-based multi-objective optimization using crowding, mutation and e-dominance, in *Evolutionary Multi-Criterion Optimization – Third International Conference* (2005)
50. M. Reyes-Sierra, C.A.C. Coello, Multi-objective particle swarm optimizers: a survey of the state-of-the-art. *Int. J. Comput. Intell. Res.* **2**, 287–308 (2006)
51. T.M. Sands, D. Tayal, M.E. Morris, S.T. Monteiro, Robust stock value prediction using support vector machines with particle swarm optimization, in *2015 IEEE Congress on Evolutionary Computation (CEC)*, May 2015, pp. 3327–3331
52. M.-P. Song, G.C. Gu. Research on particle swarm optimization: a review, in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics, 2004*, vol. 4, Aug 2004, pp. 2236–2241
53. K. Stanilov, M. Batty, Exploring the historical determinants of urban growth patterns through cellular automata. *Trans. GIS* **15**(3), 253–271 (2011)
54. S. Talatahari, 17 – optimum performance-based seismic design of frames using metaheuristic optimization algorithms, in *Metaheuristic Applications in Structures and Infrastructures*, ed. by A.H. Gandomi, X.-S. Yang, S. Talatahari, A.H. Alavi (Elsevier, Oxford, 2013), pp. 419–437
55. R.C. Team, *R: A Language and Environment for Statistical Computing* (R Foundation for Statistical Computing, Vienna, 2012). ISBN 3-900051-07-0
56. S.T. Waller, A.K. Ziliaskopoulos, A chance-constrained based stochastic dynamic traffic assignment model: analysis, formulation and solution algorithms. *Transp. Res.* **14**, 418–427 (2006)
57. S.T. Waller, K.C. Mouskos, D. Kamaryiannis, A.K. Ziliaskopoulos, A linear model for the continuous network design problem. *Comput. Aided Civ. Inf. Eng.* **21**, 334–345 (2006)
58. J. Xiao, Y. Shen, J. Ge, R. Tateishi, C. Tang, Y. Liang, Z. Huang, Evaluating urban expansion and land use change in Shijiazhuang, China, by using GIS and remote sensing. *Landscape Urban Plan.* **75**(1,2), 69–80 (2006)
59. X.-S. Yang, 1 – optimization and metaheuristic algorithms in engineering, in *Metaheuristics in Water, Geotechnical and Transport Engineering*, ed. by X.-S. Yang, A.H. Gandomi, S. Talatahari, A.H. Alavi (Elsevier, Oxford, 2013), pp. 1–23

Chapter 14

Hybrid Metaheuristic for Air Traffic Management with Uncertainty

S. Chaimatanan, D. Delahaye, and M. Mongeau

Abstract To sustain the rapidly increasing air traffic demand, the future air traffic management system will rely on a concept, called Trajectory-Based Operations (TBO), that will require aircraft to follow an assigned 4D trajectory (time-constrained trajectory) with high precision. TBO involves separating aircraft via strategic (long-term) trajectory deconfliction rather than the currently-practicing *tactical* (short-term) conflict resolution. In this context, this chapter presents a strategic trajectory planning approach aiming at minimizing the number of conflicts between aircraft trajectories for a given day. The proposed methodology allocates an alternative departure time, a horizontal flight path, and a flight level to each aircraft at a nation-wide scale.

In real-life situations, aircraft may arrive at a given position with some uncertainties on its curvilinear abscissa due to external events. To ensure robustness of the strategic trajectory plan, the aircraft arrival time to any given position will be represented here by a probabilistic distribution over its nominal assigned arrival time.

The proposed approach optimizes the 4D trajectory of each aircraft so as to minimize the probability of potential conflicts between trajectories. A hybrid-metaheuristic optimization algorithm has been developed to solve this large-scale mixed-variable optimization problem. The algorithm is implemented and tested with real air traffic data taking into account uncertainty over the French airspace for which a conflict-free and robust 4D trajectory plan is produced

Keywords Hybrid metaheuristic • Air traffic management • Optimization under uncertainty

14.1 Introduction

This section provides a brief overview of the air traffic management system and the strategic trajectory planning problem.

S. Chaimatanan (✉) • D. Delahaye • M. Mongeau
ENAC, MAIAA, Univ de Toulouse, IMT, 31400 Toulouse, France
e-mail: supatcha@recherche.enac.fr

14.1.1 Air Traffic Management: A Brief Review

Air traffic management (ATM) is a system that assists and guides aircraft from a departure aerodrome to a destination aerodrome in order to ensure its safety, while minimizing delays and airspace congestion. It manages the air traffic through the management of the three following complementary systems: airspace management (ASM), air traffic flow management (ATFM), and air traffic control (ATC).

The ASM organizes the usage of airspace. Its primary objective is to maximize the utilization of available airspace by segregating the airspace among various airspace user's needs in order to prevent interference from all users and to facilitates the flow of air traffic.

The ATFM manages the air traffic flow in order to minimize delays and prevent congestion. In Europe, this system is the concern of the Central Flow Management Unit (CFMU) of Eurocontrol. Every (non-military) operation flight performing under Instrument Flight Rules (IFR) in Europe must submit a flight plan to the CFMU. The CFMU then analyzes the compatibility of the request with the overall demand. If a request is not compatible with the airspace structure or the capacity limit, the CFMU will suggest alternative flight plan. It then distributes the accepted flight plan to all local air traffic control centers in Europe overflown by that particular flight.

The ATC then controls the air traffic in real time to ensure separation between aircraft. For this purpose, the airspace is partitioned into different airspace *sectors*, each of which is assigned to a specific group of controllers monitoring air traffic. Within each sector, a few minutes before the aircraft enters into the sector, the controllers are responsible for predicting conflicts. Then, the controllers are in charge of monitoring the traffic, maintaining aircraft separation by issuing instructions to pilots, and ensuring coordination with the neighboring sectors.

As mentioned above, in the current air traffic management system, an aircraft traveling between airports must register a flight plan in order to inform the relevant air navigation services. This flight plan includes the following information:

- Aircraft identification number, aircraft type, and navigation equipment installed on board;
- Departure airport;
- Proposed time of departure;
- Requested cruising altitude (*flight level*¹);
- Requested route of flight;
- Cruising airspeed, climb and descent profiles, and speed schedules;
- Destination airport.

¹ Flight level (FL) is a pressure altitude expressed in hundreds of feet, e.g. an altitude of 32,000 ft is referred to as FL 320.

The ATC uses this information to predict the traffic situation. It issues necessary changes to the flight plan in order to ensure aircraft separation, and to maintain the order of air traffic flow, while satisfying as much as possible the pilot's request.

Current air traffic control regulations require aircraft that operate in the en-route environment up to FL (see footnote 1) 410 to be vertically separated by at least $N_v = 1,000$ feet (ft), and to be horizontally separated by a minimum of $N_h = 5$ nautical miles (Nm). For aircraft operating above FL 410, the required minimum vertical separation is increased to $N_v = 2,000$ ft. Aircraft are considered to be *in conflict* when such a *minimum separation* requirement is violated. This conflict situation does not necessary leads to a collision; however, it is a situation that controllers must avoid. One can consider that at any given time, each aircraft has a bounded and closed reserved block of airspace defined by a three-dimensional cylinder, as shown on Fig. 14.1, in which other aircraft are not allowed to enter.

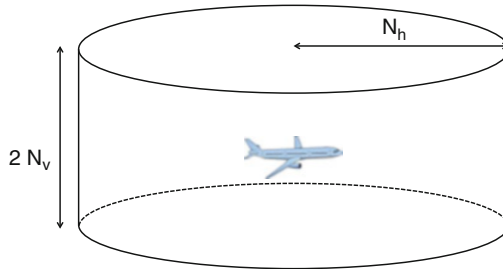


Fig. 14.1 The cylindrical *protection volume*

Because airspace, aircraft, ground systems, and human operators are limited resources which are very costly to extend, the usage of these resources has to be optimized through an effective planning. A good planning allows the ATM process to conform with the airspace user's requirements, and to be robust against unexpected events. Currently, the ATM process is performed through the following three planning phases:

- **Strategic planning.** This phase is performed from 1 year down to 1 week before real-time operations of the flights. This process aims at predicting the air traffic load, and at designing the air-route structure in order to balance capacity and demand. During this long-term and medium-term planning phase, the air traffic is macroscopically organized.
- **Pre-tactical planning.** This phase takes place from 6 days down to 1 day before the real-time operations. The objectives are to optimize the overall ATM network performance, minimizing delay and cost by fine-tuning the strategic plan using more up-to-date information of expected traffic conditions, traffic demand, available capacity and weather forecast. During this phase, the air traffic flow is not

only organized at macroscopic level, but also on each airplane. The takeoff slots² of each airplane is also managed.

- **Tactical planning.** This phase is carried out on the day of operations. Adjustments to the flight plans are performed based on the most up-to-date knowledge of the traffic situation and of the weather conditions. In this phase, individual aircraft departure slots are re-adjusted. Re-routings and alternative flight profiles can also be issued in order to avoid bottlenecks (congested sector) and to maximize airspace capacity according to real-time traffic demand. During this phase, the controller deals with the traffic inside a sector, and applies local changes to the aircraft trajectory in order to ensure aircraft separation within the corresponding sector.

14.1.2 Strategic Aircraft Trajectory Planning

In order to accommodate the increasing air traffic demand in an already saturated airspace, the world's major ATM systems (e.g. European and U.S. ATM systems) are being modernized. The Next Generation air transportation system (NextGen) is a project aiming to transform the National Airspace System (NAS) of the United States towards a satellite-based air traffic management and control system. The Single European Sky ATM Research (SESAR) project is a major collaborative project aiming at modernizing the European air traffic management system. With the soon-coming technologies that will enable more powerful communication systems, more precise surveillance systems, and more reliance automated support tools, these new ATM systems will improve safety, reduce delay and aviation pollution emissions, while maximizing the use of airspace capacity.

The new ATM systems will rely on the concept of Trajectory Based Operations (TBO) which will focus more on adapting the airspace user's demand to the current airspace capacity. The conflict detection and resolution task load will be re-distributed to the strategic planning phase. In this new ATM paradigm, an aircraft flying through the airspace will be required to follow a negotiated *conflict-free* trajectory, accurately defined in four dimensions (three spatial dimensions and time). This will significantly reduce recourse to controller's intervention during the tactical phase, thereby enabling the controllers to manage a significant increase in traffic at any given time.

In this future ATM context, the aims of the strategic aircraft trajectory planning is to reduce the number of potential conflicts between trajectories. The objective of this chapter is to present a methodology to address such a strategic planning problem given a set of flight plans for a given day at a nation-wide scale. In real-life situations, aircraft may not be able to comply with the time constraint due to external events (wind, passenger delay, etc.). Moreover, imposing hard time constraints on

² A takeoff slot is an interval of time in which the take-off has to take place.

the 4D trajectory may result in an increase of fuel consumption and aircraft engine workload, since the aircraft may have to adjust constantly its velocity. In order to improve robustness of the strategic trajectory plans and to relax the time constraints, uncertainties of aircraft arrival time to a given position is also taken into account in the trajectory optimization process.

More precisely, the given input of the strategic trajectory planning problem under consideration can be presented as follows:

- We consider a flight plan for a given day associated with a nation-wide scale airspace.
- The characteristics of the uncertainty of aircraft arrival time to any given position are given.
- For each flight, i , we suppose that the following elements are given:
 - a set of candidate routes;
 - a set of candidate flight levels;
 - a set of candidate departure times;

In the sequel, we shall often refer to flight i as *trajectory i* , or even *aircraft i* . The proposed strategic planning methodology consists of four main modules: a 4D-trajectory generator, a conflict-detection module, an interaction evaluation module, and a hybrid-metaheuristic optimization module (Fig. 14.2). The 4D-trajectory generator is used to provide a 4D trajectory given an alternative route, an alternative flight level, and an alternative departure time. Then, the probabilistic conflict-detection module computes the probability of conflict involving a given 4D trajectory. After that, the interaction evaluation module will compute the level of interaction between trajectories at a nation-wide scale. The hybrid-metaheuristic optimization algorithm manages the search of an optimal set of alternative routes, alternative flight levels, and alternative departure times that minimize the potential conflicts (or the *interaction*-defined later) between trajectories.

The optimal solution obtained is based on the following assumptions and simplifications:

- The airspace is considered as a Euclidean space. Latitudes and longitudes on the Earth's surface are transformed into (x, y) coordinates.
- The altitude, in feet, will be represented by the z coordinate.
- Each given initial route is a straight line from the departure airport to the destination airport.
- Aircraft speed is assumed to be changing only linearly between two consecutive sampling time steps.
- Uncertainty of aircraft arrival time does not grow with time.
- Wind conditions and weather forecast are not taken into account in the trajectory optimization process.

The remaining parts of this chapter are organized as follows. In Sect. 14.2, previous works related to air traffic management problems are discussed. Section 14.3 presents the mathematical model of the strategic trajectory planning problem. Section 14.4 presents a methodology to detect conflicts, and to compute conflict probability between aircraft trajectories. A hybrid-metaheuristic method designed to solve

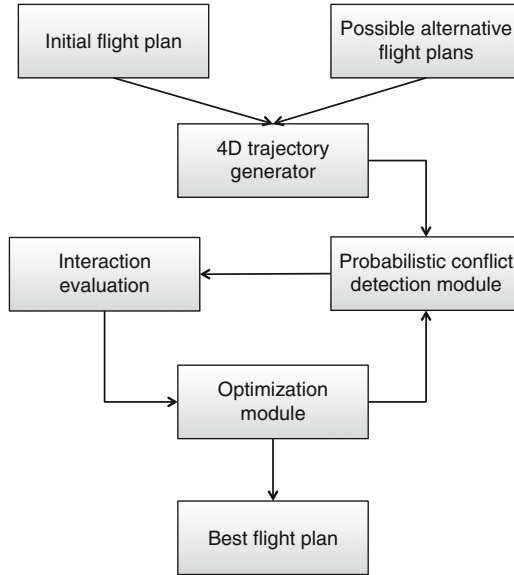


Fig. 14.2 Strategic trajectory planning procedure

the strategic trajectory planning problem aiming at minimizing the total interaction between aircraft trajectories is presented in Sect. 14.5. Computational experiments with the proposed strategic trajectory planning methodology are presented in Sect. 14.6. Conclusions and perspectives are discussed in Sect. 14.7.

14.2 Previous Related Works

Over the last decades, numerous researches on the air traffic management problem have been conducted. We refer the reader interested by a survey on modeling and optimization in air traffic to the recent book [12]. A survey on mathematical optimization models for air traffic management problems based on different air traffic management strategies is provided in [1]. A comparison of different optimization methods (deterministic and metaheuristic optimization approaches) used for air traffic management is provided in [18].

In the strategic planning framework, aircraft trajectories can be separated in many different ways. One of the simplest and the most used method is to modify the departure time of aircraft. This is commonly referred to as *ground delay* or *ground holding*. The main idea of the ground holding strategy is to limit the number of airborne aircraft at any given time. Examples of works related to ground holding are [3, 19], and [24]. Delaying aircraft on the ground is effective since it prevents aircraft from flying extra distance to avoid congested areas or flying in a holding pattern around congested airport, which induce extra fuel consumption. However, with increasing air traffic demand, significant delays still have to be assigned to a

large number of aircraft in order to meet all airspace-sector and airport capacity constraints. Besides, the ground holding strategy is more effective for the situation where congestion is likely to occur at the airports, which is not the case in Europe where most congestion occurs in the airspace sectors.

In [8] and [11], another idea to separate trajectories is presented based on speed regulations. Speed regulations introduce additional degrees of freedom to manage the flow of air traffic. However, it is effective at the fine-grain level which is irrelevant in the strategic trajectory planning context where there remain a high level of uncertainty. Furthermore, it requires numerous extensive and fine-tuned computations, which is not viable for a large-scale problem.

Other commonly-used strategies consider diverting the flight (re-routing), or modifying the flight levels, or a combination of the above-mentioned methods. To simplify the problem, several works rely on a flow-based air traffic model, where aircraft trajectories are grouped into several flows. For instance, [15] addresses large-scale (1 day traffic over France) air traffic flow problems via a flow-based trajectory allocation, where the optimal separated 3D trajectory are obtained using an A* algorithm or using a genetic algorithm (GA) global search strategy. In [4], a ground holding is assigned to each aircraft and an optimal flight level is subsequently allocated to each flow of aircraft using constraint programming. Their results show that such rerouting and flight level re-allocation yield decrease in delays. In spite of the fact that the flow-based air traffic model has advantages in terms of reduced computation time, it cannot separate aircraft that belong to the same flow of trajectories.

To consider now each flight *individually*, air traffic flow models can rely on a collection of subgraphs, whose the nodes represent the airports and waypoints over-flown by each flight, and whose arcs connect the nodes for each flight. For instance, in [5], the authors show that the departure-time and alternative-route allocation problem is NP hard. Their optimal ground-holding times and alternative routes are obtained by solving a 0-1 integer model taking into account airspace sector capacity. Their models were implemented and tested with realistic datasets consisting of 2–6 airports. In [6, 7], integer optimization approaches are used in order to allocate ground delays and rerouting options to trajectories taking into account airspace sector capacity constraints. Thus, [6, 7, 5] propose improvements of air traffic at the airspace sector level but do not manage conflicts.

The authors of [2], introduce a mixed-integer programming model to minimize traveling time, operating/fuel cost, air/sound pollutions under separation and technical constraints. The optimal arcs and nodes (in a 3D-mesh network), speeds, and departure/arrival times for each flight are obtained by an exact deterministic method. However, the approach was tested on instances limited to problems involving ten flights.

Reference [21, 20], the authors focus on managing each individual trajectory in large problems. Congestion in the airspace sectors is minimized by allocating to each flight optimal departure times and alternative routes (based on route-beacons navigation) using genetic algorithms (GA). Their results show that GA is very efficient in solving highly complex problems. Nevertheless, GA is not well adapted for the large-scale 4D trajectory planning problems that we are considering, due

to excessive memory requirement intrinsic to population-based optimization algorithms whose performance depends on the size of population.

In the future ATM context, aircraft trajectory can be represented by a time sequence of 4D coordinates. In [9, 10], preliminary studies on the optimization of individual 4D trajectories are presented. In these papers, optimal (conflict-free) 4D trajectories for individual flights are allocated by solving a combinatorial optimization problem using a non-population-based hybrid-metaheuristic optimization method. The numerical results presented in [10] show advantages of the hybrid-metaheuristic optimization approach on ATFM problems. However, the discretization of the search domain (candidate departure times and trajectories) induces high combinatorics.

Uncertainties of aircraft position were taken into account in the conflict detection and resolution problem addressed in the work presented, for example, in [14, 16, 22]. Aircraft positions are modeled as a probabilistic distribution, then the predicted aircraft positions are computed over a certain time window using a dynamic model of aircraft, and conflict probabilities are evaluated. These methods are suitable for mid-term and short-term conflict-detection and resolution problems involving a small number of aircraft. However, they are not suitable for the large-scale problems that we are attempting to address in this work, due to the heavy computational burden implicated in predicting aircraft positions using aircraft dynamic models. We refer the reader interested by a review of conflict-detection and resolution modeling methods to [17].

In this chapter, we put forward the work presented in [10], by relaxing the solution space and proposing an alternative, mixed-integer programming formulation of the problem. Moreover, we introduce a methodology to take into account uncertainty of aircraft trajectories in the strategic trajectory planning problem. We also propose new intensification local-search steps, and we describe a computationally-efficient hash-table based method for detecting and evaluating probabilities of conflict between trajectories. Finally, we prove the viability of the overall methodology on large-scale air traffic data on the French airspace.

14.3 Mathematical Model

This section set the mathematical framework of the proposed strategic trajectory planning methodology. First, the assumed uncertainty on aircraft trajectory is characterized. Then, methods that are used to separate the aircraft trajectories are described. Finally, a concept of interaction between trajectories, and a mathematical formulation of the strategic trajectory planning problem under the form of a mixed-integer optimization problem are presented.

14.3.1 Uncertainty

Conflict detection methods can be roughly classified into three categories [16]: nominal, worst-case, and probabilistic conflict detections, according to the assumptions made on the predicted aircraft trajectory. The *nominal* conflict detection does not take into account deviation of aircraft from its assigned (nominal) trajectory. The *worst-case* conflict detection identifies the conflict as a situation in which the distance between the *envelopes* of the predicted trajectories (the set of all possible trajectories) is less than the minimum separation requirements. The *probabilistic* conflict detection method involves computing probability of conflict between aircraft whose trajectories are described with probability density functions. In other words, it computes the probability that two aircraft will penetrate into the (cylindrical) protection volume of one another. It is suitable for assessing the air traffic condition in a large-scale traffic scenario with high level of uncertainties, for example in strategic trajectory planning.

We shall use in the remaining of this chapter the notation $P_i = (x_{P_i}, y_{P_i}, z_{P_i}, t_{P_i})$ to designate a 4D point on trajectory i . We shall call its fourth coordinate, t_{P_i} , the *assigned arrival time* of aircraft i at the point $(x_{P_i}, y_{P_i}, z_{P_i})$. Let us consider two trajectories, A and B, and let us first consider the case where time uncertainty is *not* taken into account. In the absence of time uncertainty, when the horizontal separation, $d_h = \sqrt{(x_{P_A} - x_{P_B})^2 + (y_{P_A} - y_{P_B})^2}$, is less than 5 Nm, and when the vertical separation, $d_v = |z_{P_A} - z_{P_B}|$, is less than 1,000 ft, the arrival times of both aircraft must be separated in time, i.e. $d_t = |t_{P_A} - t_{P_B}|$, strictly greater than zero.

In general, an aircraft is able to follow a given flight profile with very high accuracy thanks to the flight management system (FMS³). The residue uncertainty of aircraft position is more likely to occur in the time domain. The aircraft may arrive at a given position with a time error due to, for example, wind conditions, external temperature, aircraft weight estimation errors, passenger delay, etc.

Consider now the case with time uncertainty: let t_ε be the *maximum time error* (defined by the user). The *predicted* arrival time of an aircraft at a position P under uncertainty therefore lies in the interval:

$$[t_P - t_\varepsilon, t_P + t_\varepsilon].$$

For the purpose of potential conflict detection, we assume that the predicted aircraft arrival time can be modeled as a random variable with the following triangular distribution defined over the interval $[t_P - t_\varepsilon, t_P + t_\varepsilon]$. Given a lower limit $t_P - t_\varepsilon$, an upper limit $t_P + t_\varepsilon$, the *predicted* arrival time, \hat{t}_P , to the position P is given by the probability density function:

$$\hat{t}_P(t) = \mathcal{T}_{P, t_\varepsilon}(t),$$

where $\mathcal{T}_{P, t_\varepsilon}(t)$ denotes the triangular distribution:

³ Flight management system (FMS) is an on-board computer system that determines the aircraft exact position and calculates the lateral and horizontal guidance for the aircraft.

$$\mathcal{T}_{P,t_\epsilon}(t) = \begin{cases} 0 & \text{for } t < t_P - t_\epsilon, \\ \frac{(t - t_P + t_\epsilon)}{t_\epsilon^2} & \text{for } t_P - t_\epsilon \leq t \leq t_P, \\ \frac{(t_P + t_\epsilon - t)}{t_\epsilon^2} & \text{for } t_P < t \leq t_P + t_\epsilon, \\ 0 & \text{for } t_P + t_\epsilon < t. \end{cases}$$

To explain the process to detect conflicts between two aircraft trajectories in our triangular distribution case, let us first consider two trajectories A and B illustrated in Fig. 14.3. For simplicity, let us first assume that trajectories A and B are defined by continuous functions, and let P_A and P_B be a pair of any points on the trajectories A and B respectively. To identify conflict between these two trajectories, we must check the minimum separation between all possible pairs of points P_A and P_B (pair-wise comparison). The predicted arrival time, \hat{t}_{P_A} , of aircraft A to the given point P_A ,

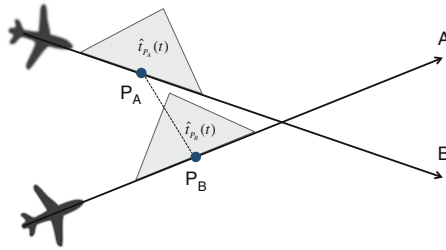


Fig. 14.3 Measuring conflict between two continuous trajectories A and B

and the predicted arrival time, \hat{t}_{P_B} , to the given point P_B are given by:

$$\hat{t}_{P_A}(t) = \mathcal{T}_{t_{P_A}, t_\epsilon}(t),$$

and

$$\hat{t}_{P_B}(t) = \mathcal{T}_{t_{P_B}, t_\epsilon}(t).$$

A potential conflict between trajectories A and B occurs when there exists a pair of points, P_A and P_B , from each trajectory such that

$$d_h = \sqrt{(x_{P_A} - x_{P_B})^2 + (y_{P_A} - y_{P_B})^2}$$

is less than 5 Nm, $d_v = |z_{P_A} - z_{P_B}|$ is less than 1,000 ft, and the intersection between intervals $[t_{P_A} - t_\epsilon, t_{P_A} + t_\epsilon]$ and $[t_{P_B} - t_\epsilon, t_{P_B} + t_\epsilon]$ is not empty. The conflict probability, denoted $\mathcal{P}_C(P_A, P_B)$, between the point P_A and point P_B can be computed from:

$$\mathcal{P}_C(P_A, P_B) = \int_{t_{start}}^{t_{end}} \hat{t}_{P_A} \hat{t}_{P_B} dt, \tag{14.1}$$

where t_{start} and t_{end} are respectively the lower and upper bounds of the interval $[t_{P_A} - t_\epsilon, t_{P_A} + t_\epsilon] \cap [t_{P_B} - t_\epsilon, t_{P_B} + t_\epsilon]$.

To implement our conflict detection algorithm, we discretize the 4D trajectories. The sampling time step, t_s , must be set (by the user) sufficiently small to guarantee that any conflict occurring between two consecutive sampling steps will be detected. Doing so, trajectories A and B can be represented respectively, by the time sequences of 4D coordinates $\{P_{A,k_A}\}_{k_A=1}^{K_A}$, and $\{P_{B,k_B}\}_{k_B=1}^{K_B}$, where K_A and K_B are the number of sampling points corresponding to trajectories A and B respectively (see Fig. 14.4). In order to detect the conflicts between the two trajectories, we must verify the minimum separation constraint between every possible pair of sampled points P_{A,k_A} and P_{B,k_B} .

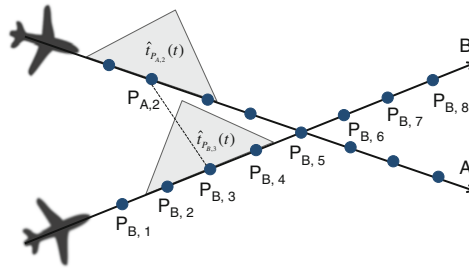


Fig. 14.4 Measuring conflict between two discretized trajectories A and B

The conflict probability, $\mathcal{P}_C(P_{A,k_A}, P_{B,k_B})$, associated to every pair of sample points P_{A,k_A} and P_{B,k_B} of trajectories A and B. This can be computed using Eq. (14.1). However, this pair-wise comparison is time consuming. It requires prohibitive time in a large-scale application context as the one considered in this study. A fast algorithm to detect such a probabilistic violation of the minimum separation requirements (i.e. to compute the conflict probabilities associated to all pairs of sampled trajectory points between large-scale aircraft trajectories) will be presented in Sect. 14.4.

14.3.2 Trajectory Separation Methods

In this subsection, we describe the three possible trajectory separation methods we are considering in order to avoid conflicts:

- shifting the departure time,
- changing the flight level,
- modifying the route (horizontal flight profile).

The alternative departure time, alternative flight level, and alternative route to be allocated to each flight are modeled as follows.

14.3.2.1 Alternative Departure Time

The departure time of each flight, i , can be shifted by a positive (delay) or a negative (advance) time shift denote δ_i . The departure time, t_i , of flight i is therefore

$$t_i = t_{i,0} + \delta_i,$$

where $t_{i,0}$ is the initially-planned departure time of flight i . Following common practice in airports, the set of possible values for δ_i will be discrete.

14.3.2.2 Alternative Flight Level

To separate the trajectories in the vertical plane, we define another decision variable associated to each flight i : a flight level shift $l_i \in \mathbb{Z}$. Therefore, the flight level of each flight i is given by:

$$FL_i = FL_{i,0} + l_i,$$

where $FL_{i,0}$ is the initially-planned flight level of flight i . Figure 14.5 shows a trajectory with two alternative flight levels.

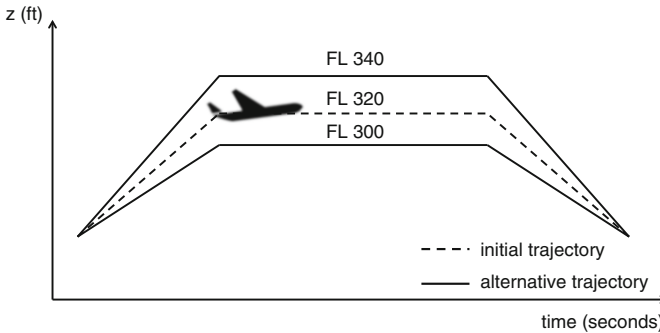


Fig. 14.5 Two alternative vertical profiles of a trajectory (two alternative flight levels)

14.3.2.3 Alternative Route Design

An alternative route should not deviate too much from the nominal route. It should also be computed in a short computation time. To generate an alternative route, we modify the given initial *horizontal flight profile* of a trajectory, i , by placing a set of virtual waypoints near the initial horizontal flight profile of flight i , and then by reconnecting the successive waypoints with straight-line segments.

We call *longitudinal axis* (x') the axis that is tangent to the initial en-route segment, and the *lateral axis* (y') is the axis that is perpendicular to the longitudinal axis. The position of each waypoint will be defined using these relative $x'y'$ -reference axes.

We define, for each flight i , a vector, w_i , of virtual waypoints (optimization variables) used to control the trajectory shape of flight i : $w_i = (w_i^1, w_i^2, \dots, w_i^m)$, where M denotes the number of virtual waypoints that the user is allowed to introduce, and where $w_i^m = (w_{ix'}^m, w_{iy'}^m)$ is the m th virtual waypoint of trajectory i , where $w_{ix'}^m$ and $w_{iy'}^m$ are the longitudinal and lateral components of w_i^m respectively. Figure 14.6, illustrates possible alternative horizontal profiles for a given trajectory constructed with $M = 2$ virtual waypoints.

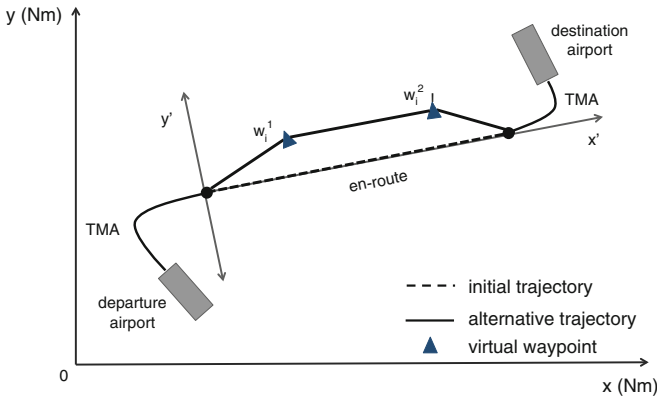


Fig. 14.6 An alternative horizontal profiles for a given trajectory, i , constructed with $M = 2$ virtual waypoints

Remark that such an alternative trajectory is likely to yield an increase in flight duration when compared with the initial trajectory. To compensate this increased flight duration, the altitude profile will be updated to avoid a premature descent. Let T_{ext} be the increased flight duration of flight i . In the case of a regional flight, whose flight phases are all carried out in the same (current) airspace sector, the altitude profile is updated by extending the cruise phase at the top of descent for a duration of T_{ext} as illustrated in Fig. 14.7.

On the other hand, for a flight whose origin or destination airports are outside of the current airspace, the top of descent of such flight may not be in the current airspace sector. Therefore, we update the altitude profile by extending the flight at maximum altitude (in the current airspace) for a duration T_{ext} . In this case, the vertical profile is updated according to six possible cases according to whether the origin/destination airports are in the current airspace or not and to whether the initial trajectory has a cruise (constant-level) phase or not, as illustrated in Fig. 14.8.

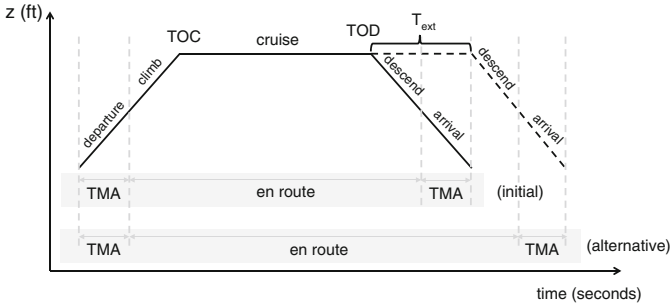


Fig. 14.7 Altitude-profile update: extending cruise phase at the top of descent (TOD)

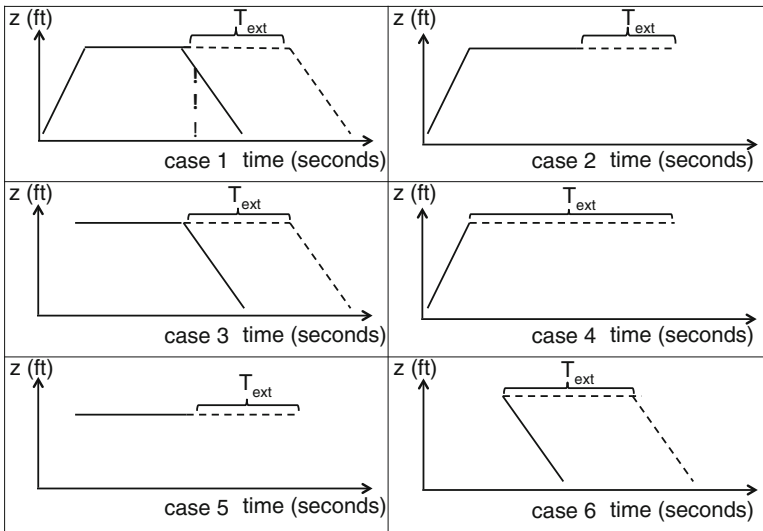


Fig. 14.8 Altitude profile update: six possible ways to extend the trajectory at maximum altitude

14.3.3 Optimization Formulation

In this subsection, we present an optimization formulation of the strategic 4D trajectory planning problem. The strategic 4D trajectory planning methodology using route/flight-level/departure-time allocation can be formulated as an optimization problem attempting at minimizing the interaction between trajectories.

14.3.3.1 Given Data

A problem instance is given by:

- A set of N initial (nominal) discretized 4D trajectories;
- The maximal time error, t_{ϵ} ;
- The sampling time step: t_s ;

- The interpolation sampling time step: t_{interp} ;
- The number of allowed virtual waypoints: M ;
- The discretization time step for the possible delay/advance departure-time shift interval: δ_s ;
- For each flight i , for $i = 1, \dots, N$:
 - The initial planned departure time: $t_{i,0}$;
 - The maximum allowed advance departure time shift: $\delta_a^i < 0$;
 - The maximum allowed delay departure time shift: $\delta_d^i > 0$;
 - The initial planned flight level: $FL_{i,0}$;
 - The maximum allowed flight level shift: $l_{i,max}$;
 - The length of the initial planned route: $L_{i,0}$;
 - The maximum allowed route length extension coefficient: $0 \leq d_i \leq 1$;
 - The user-defined parameters controlling the dimensions of the feasible domains for placing the virtual waypoints: a_i and b_i .

14.3.3.2 Decision Variables

As mentioned above, we consider three ways to separate trajectories. In the time domain, one can use a departure-time shift δ_i is associated to each flight, i . In the 3D space, one can rely on a vector, w_i , of virtual waypoint locations, $w_i = (w_i^1, w_i^2, \dots, w_i^M)$ associated to each flight, i , where M is the number of virtual waypoints. Finally, in the vertical plane, a flight-level shift, l_i may be applied to each flight i .

Let us set the compact vector notation:

$$\boldsymbol{\delta} = (\delta_1, \delta_2, \dots, \delta_N),$$

$$\mathbf{l} = (l_1, l_2, \dots, l_N).$$

and

$$\mathbf{w} = (w_1, w_2, \dots, w_N).$$

Therefore, the decision variables of our route/departure-time allocation problem can be represented by the vector:

$$u := (\boldsymbol{\delta}, \mathbf{l}, \mathbf{w}).$$

We shall denote by u_i the components of u . It is a vector whose components are related to the modification of the i th trajectory, thereby:

$$u_i := (\delta_i, l_i, w_i)$$

14.3.3.3 Constraints

The above optimization variables must satisfy the following constraints:

Allowed departure time shift. The departure time of flight i is given by an auxiliary optimization variable, t_i , which is directly linked to the above decision variables as follows:

$$t_i = t_{i,0} + \delta_i,$$

where $t_{i,0}$ is the initial planned departure time of flight i .

In practical problems, passengers may have to transfer from one flight to another in order to get to their final destination. This generates precedence constraints stipulating that certain flights must arrive at the airport before the departure of others. In addition, each aircraft may fly several flights a day. This raises a constraint of minimum *rotation time* between flights (time required to disembark the passengers, to service the aircraft, and to embark passengers for the next flight). These constraints are not taken into account in this work; however they can easily be handled by pre-processing the set of feasible time shifts of each flight.

In order to prevent excessive delay (or advance) of departure time, the departure-time shift δ_i is limited to lie in the interval

$$[\delta_a^i, \delta_d^i].$$

However, common practice in airports conducted us to rely on a discretization of this time interval. Given a user-defined departure time shift step-size δ_s (to be set by the user), this yields $N_a^i := \frac{-\delta_a^i}{\delta_s}$ possible advance slots and $N_d^i := \frac{\delta_d^i}{\delta_s}$ possible delay slots of flight i . Parameters are to be set by the user so that both δ_a^i and δ_d^i are multiples of δ_s . Therefore, the set, Δ_i , of all possible departure time shifts of flight i :

$$\Delta_i := \{-N_a^i \cdot \delta_s, -(N_a^i - 1) \cdot \delta_s, \dots, -\delta_s, 0, \delta_s, \dots, (N_d^i - 1) \cdot \delta_s, N_d^i \cdot \delta_s\}. \quad (14.2)$$

Maximum allowed flight-level changes. In order to limit the change of flight levels, the flight level shift is also bounded. The set, ΔFL_i , of all possible flight-level shifts of flight i is given by:

$$\Delta FL_i = [FL_{i,0} - l_{i,max}, \dots, FL_{i,0} - 1, 0, FL_{i,0} + 1, \dots, FL_{i,0} + l_{i,max}]. \quad (14.3)$$

Maximal route length extension. The alternative trajectory induces route length extension which causes an increase of fuel consumption. Therefore, it should be limited so that it remains acceptable by the airline. Let $0 \leq d_i \leq 1$ be the maximum allowed route length extension coefficient of flight i (to be set by the user). The alternative en-route profile of flight i must satisfy:

$$L_i(w_i) \leq (1 + d_i)L_{i,0}, \quad (14.4)$$

where $L_i(w_i)$ is the length of the alternative en-route profile determined by w_i . This constraint can be satisfied a priori simply by restricting the set of possible waypoint locations (as will be described below).

Allowed waypoint locations. To limit the search space, to prevent undesirable sharp turns, and to restrain the route length extension, we bound the possible location of each virtual waypoint. For simplicity, for each trajectory i , and for each waypoint w_i^m , its longitudinal component, $w_{ix'}^m$, is set to lie in the interval:

$$W_{ix'}^m := \left[\left(\frac{m}{1+M} - b_i \right) L_{i,0}, \left(\frac{m}{1+M} + b_i \right) L_{i,0} \right], \quad (14.5)$$

for $m = 1, \dots, M$; and $i = 1, \dots, N$, and where $b_i \geq 0$ is a user-defined coefficient controlling the length of the interval $W_{ix'}^m$.

Furthermore, to avoid sharp turns, the longitudinal position of the virtual waypoints should not be too close to each other. To obtain a regular trajectory, the longitudinal component of two adjacent waypoints must not overlap, i.e.

$$\left(\frac{m}{1+M} + b_i \right) < \left(\frac{m+1}{1+M} - b_i \right),$$

and hence the user should choose parameter b_i so that

$$b_i < \frac{1}{2(M+1)}.$$

Similarly, the lateral component, $w_{iy'}^m$, is restricted to lie in the interval:

$$W_{iy'}^m := [-a_i \cdot L_{i,0}, a_i \cdot L_{i,0}], \quad (14.6)$$

where $a_i \leq 0$ is a user-defined coefficient. The box-size parameters $(a_i, b_i)_{i=1}^N$ should be chosen so that the maximal route length extension constraint (14.4) is satisfied for all possible locations of the m waypoints in the 2D boxes $\{W_{ix'}^m \times W_{iy'}^m\}_{m=1}^M$, for every trajectory i ($i = 1, 2, \dots, N$). Figure 14.9 illustrates for a trajectory i the 2D boxes of possible locations for $M = 2$ virtual waypoints, an example of positions for w_i^1 and w_i^2 , and the resulting alternative trajectory. More precisely, for each trajectory i , a_i and b_i must be chosen a priori so that:

$$\max_{w_i} \{L_i(w_i) | w_i \in W_{ix'} \times W_{iy'}\} \leq (1 + d_i)L_{i,0}$$

14.3.3.4 Objective Function

In the *strategic* trajectory planning, where uncertainty is too large to fine-tune the trajectories, we focus on separating roughly aircraft trajectories rather than on solving precisely each conflict locally. Therefore, we introduce here the concept of

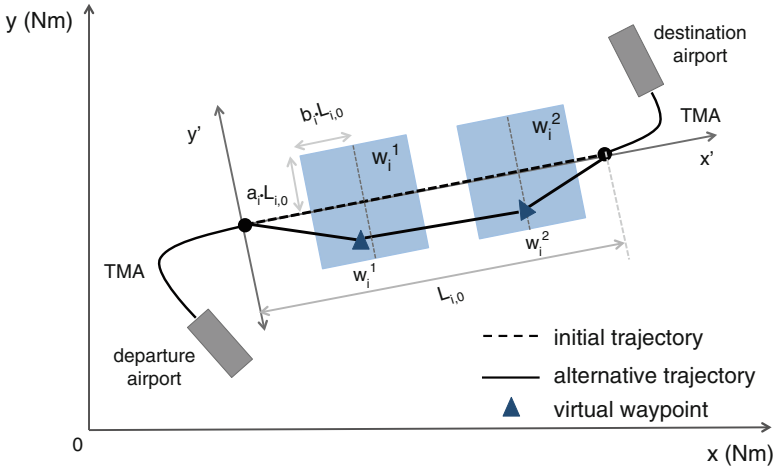


Fig. 14.9 2D boxes of possible locations for $M = 2$ virtual waypoints, a proposition of location of the virtual waypoints, and the corresponding alternative trajectory

interaction between trajectories to define a situation that occurs in the planning phase, when more than one trajectory compete for the “same space” at the “same period of time”.

For given values of the decision variables $u = (\delta, \mathbf{l}, \mathbf{w})$, one must first discretize each of the N resulting alternative trajectories into a sequence of 4D points: $\{P_{i,k}(u_i)\}_{k=1}^{K_i}$, $i = 1, 2, \dots, N$. Each of these points depends only on the i th component of u .

Let us define an *interaction at a point* $P_{i,k}(u_i)$ to be the sum of all the conflict probabilities associated to point $P_{i,k}(u_i)$; we denote it $\Phi_{i,k}(u)$. Remark that it depends also on the other trajectories $j \neq i$. Hence,

$$\Phi_{i,k}(u) = \sum_{i=1}^N \sum_{\substack{j=1 \\ j \neq i}}^N \sum_{k=1}^{K_i} \sum_{l=1}^{K_j} \mathcal{P}_C(P_{i,k}(u_i)P_{j,l}(u_j)),$$

where K_i and K_j are the number of sampling points for trajectory i and j respectively.

The *interaction associated with trajectory* i , denoted $\Phi_i(u)$, is defined as follows:

$$\Phi_i(u) = \sum_{k=1}^{K_i} \Phi_{i,k}(u).$$

Finally, the *total interaction between trajectories*, $\Phi_{tot}(u)$, for a whole N -aircraft traffic situation is simply defined as:

$$\Phi_{tot}(u) = \sum_{i=1}^N \Phi_i(u) = \sum_{i=1}^N \sum_{k=1}^{K_i} \Phi_{i,k}(u). \tag{14.7}$$

One wishes to determine values for the optimization variables δ_i , l_i , and w_i for each flight $i = 1, 2, \dots, N$ so as to minimize the total interaction, $\Phi_{tot}(u)$, between the N given trajectories.

To summarize, the strategic trajectory planning problem with uncertainty can be represented by an interaction minimization problem formulated as a mixed-integer optimization problem as follows:

$$\begin{aligned} & \min_u \Phi_{tot}(u) \\ & \text{subject to} \\ & \delta_i \in \Delta_i, \quad i = 1, 2, \dots, N \\ & l_i \in \Delta FL_i, \quad i = 1, 2, \dots, N \\ & w_i^m \in W_{ix'}^m \times W_{iy'}^m, \quad m = 1, 2, \dots, M, \quad i = 1, 2, \dots, N, \end{aligned} \tag{P1}$$

where $\Phi_{tot}(u)$ is defined by (14.7), and Δ_i , ΔFL_i , $W_{ix'}^m$, and $W_{iy'}^m$ are defined by (14.2), (14.3), (14.5), and (14.6) respectively.

The optimization formulation (P1) involves mixed-integer variables introducing high combinatorics to the search space. We have $W_{ix'}^m \times W_{iy'}^m \subseteq \mathbb{R}^2$ for $m = 1, 2, \dots, M$ and $i = 1, 2, \dots, N$, and therefore $\mathbf{w} \in \mathbb{R}^{2MN}$. Each discrete departure-time shift-variable (δ_i) feasible set has cardinality $|\Delta_i| = \frac{|\delta_a^i| + |\delta_d^i|}{\delta_s} + 1$, which implies $\delta \in \mathbb{Z}^{(N \frac{|\delta_a^i| + |\delta_d^i|}{\delta_s} + 1)}$. Finally, each discrete flight-level shift variable (l_i) feasible set has cardinality $2l_{i,max} + 1$; therefore we have $\mathbf{l} \in \mathbb{Z}^{N(2l_{i,max} + 1)}$.

We emphasize the fact that one evaluation of the objective function Φ_{tot} for one proposition of the decision variables $u = (\delta, \mathbf{l}, \mathbf{w})$ involves discretizing each of the N resulting candidate trajectories. Remark also that, the objective function of problem (P1) is non-separable, because each term $\Phi_{i,k}(u)$ does not depend solely on the variable u_i ; it is also affected by neighboring trajectories. The evaluation of the objective function involves a heavy computational burden in practice, as this will be seen in the sequel of this chapter where we consider a real-world problem at the nation-wide scale. Besides, the objective function may feature several local optima (the objective function may easily be shown to be multimodal). This route/flight-level/departure-time assignment problem with uncertainty is therefore sufficiently difficult to motivate recourse to a stochastic method of optimization.

14.4 Interaction Detection Module

To evaluate the objective function, Φ_{tot} , at a candidate solution, u , one needs to compute the interaction at each possible pair of sampled trajectory points involved in N aircraft trajectories. To avoid the exhaustive $\frac{N(N-1)}{2}$ time-consuming pair-wise comparisons, which is prohibitive in our large-scale application context, we propose the following grid-based interaction-detection scheme.

First, we define a four-dimensional (3D space + time) *grid*. The dimension of this 4D grid must be large enough to include the N given trajectories (and all its possible modifications through our decision variables). For instance, the time dimension of the grid must span enough to include the earliest and the latest flights on a given operational day taking into account all candidate departure-time shift options.

The 4D grid is partitioned into *cells* (see Fig. 14.10). To detect interactions, the idea is to store the N trajectories in each corresponding cell in the 4D grid. Then for each trajectory i , and for each cell (I_x, I_y, I_z, I_t) corresponding to each sampling point $P_{i,k} := (x_{P_{i,k}}, y_{P_{i,k}}, z_{P_{i,k}}, t_{P_{i,k}})$, we simply need to check all the surrounding cells corresponding to the time period $[t_{P_{i,k}} - 2t_\epsilon, P_{i,k} + 2t_\epsilon]$. If these neighboring cells are occupied by other aircraft, for instance j , we then note $j \in (I_x, I_y, I_z, I_t)$, and then the conflict probabilities between point $P_{i,k}$ and the sample point corresponding to those aircraft are computed (otherwise, it is null).

In order to optimize the required computation memory, we implement the interaction-detection scheme using a so-called *harsh table*, which is a data structure that maps *keys* to values or *entries*. It allows us to store information in an array without the need to define a priori the size of the array. Moreover, the hash table only stores data as it is created; therefore it does not use memory for the (very numerous) empty cells in the array.

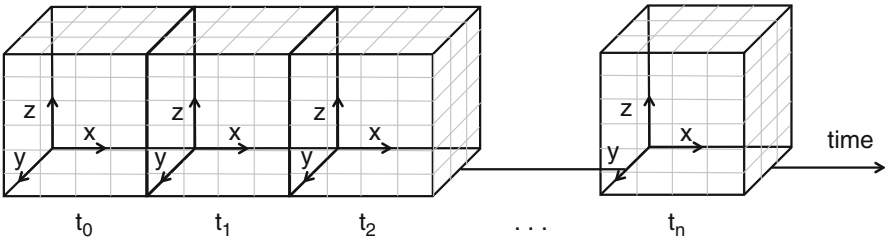


Fig. 14.10 Four dimension (space–time) grid

In order not to underestimate interaction (missing the loss of spatial separation occurring between two successive sampling time steps), trajectories must be discretized with a sufficiently-small sampling time step, t_s , which depends on the maximum possible aircraft horizontal and vertical speeds. Figure 14.11 illustrates an undetected violation of the horizontal minimum separation occurring between two successive time steps. As stated in [4], the worst-case scenario for interaction detection in the horizontal plane occurs when two aircraft follow parallel trajectories that are separated by a distance, D , less than or equal to the horizontal separation norm, $N_h = 5 Nm$, at maximum horizontal speed, V_{hmax} , with headings in opposite directions. Hence, in the horizontal plane, undetected interaction can occur when:

$$t_s > \frac{N_h}{V_{hmax}} \cos \left(\arcsin \left(\frac{D}{N_h} \right) \right).$$

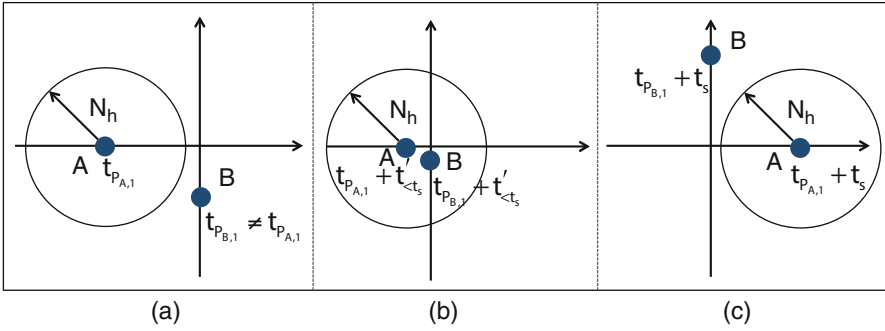


Fig. 14.11 Undetected violation of horizontal separation when the sampling step, t_s , is too large. In (a) and (c) the horizontal spatial distances between A and B are larger than $N_h = 5$ Nm, therefore the violation of the horizontal minimum separation, which occurs in the meantime (figure (b)) cannot be detected

In the vertical plane, the worst-case scenario occurs when one aircraft is climbing at a maximum rate of climb, RoC_{max} , and another is descending at maximum rate of descent, RoD_{max} (see Fig. 14.12). Thus, in the vertical plane, in an analogical way as what was done in [4] for the horizontal plane, we can easily show that undetected interaction can occur when:

$$t_s > \frac{N_v}{(RoC_{max} + RoD_{max})},$$

where $N_v := 1,000$ ft is the vertical separation norm.

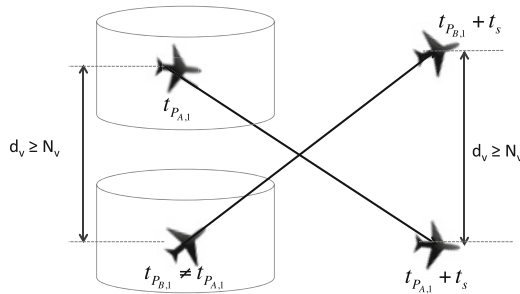


Fig. 14.12 Undetected violation of the vertical minimum separation between two aircraft when the sampling step, t_s , is too large

In order to avoid such undetected conflicts, one can therefore simply choose a sufficiently small value for the (user-provided) sampling time step, t_s . However, using too small sampling time step leads to a large number of trajectory sample points, which in turn requires more computation time and memory. Instead, we propose an

inner-loop algorithm, called `Interp`, detecting the violation of minimum separation requirements between two sampling times, t and $t + t_s$, by *interpolating* aircraft positions with a sufficiently small interpolation step size, t_{interp} . This t_{interp} value must be set by the user so as to guarantee that no interaction remains undetected. Then, one checks each pair of these interpolated points. The algorithm stops when a violation of the minimum separation requirements is identified or when every pair of points have been checked. The inner-loop interpolation algorithm called `Interp` is described in Fig. 14.13. The algorithm to compute the total interaction between the N trajectories, $\Phi_{tot}(\mathbf{u})$, is described in detail in Fig. 14.14.

Algorithm `Interp`

Require: $P_{i,k}, P_{j,l}$

- 1: Discretize, using time step t_{interp} , the trajectory segment $[P_{i,k}, P_{i,k+1}]$ and $[P_{j,l}, P_{j,l+1}]$ as $\{P_\alpha\}_{\alpha=1}^K$ and $\{Q_\beta\}_{\beta=1}^K$ respectively;
 - 2: **for** $k = 0 \rightarrow K$ **do** ▷ for each pair of interpolated points
 - 3: Initialize $\mathcal{P}_C := 0$;
 - 4: Compute conflict probability, $\mathcal{P}_C := \mathcal{P}_C(P_k, Q_k)$ using (14.1);
 - 5: **if** $\mathcal{P}_C > 0$ **then**
 - 6: Return \mathcal{P}_C ;
 - 7: **End**;
 - 8: **end if**
 - 9: **end for**
 - 10: Return \mathcal{P}_C ;
-

Fig. 14.13 Inner-loop interpolation algorithm

14.5 Hybrid-Metaheuristic for Strategic Trajectory Planning

In our country-wide air traffic scale application context, the evaluation of the objective function value relies on a black-box simulation through the interaction detection scheme introduced in the previous section. For such a large-scale problem, this simulation requires a very large computation memory. Therefore, to solve the interaction minimization problem (P1), we rely on a non-population based hybrid-metaheuristic optimization method that combines the advantages of simulated annealing and of an iterative-improvement local search.

After presenting brief overviews of simulated annealing and iterative-improvement local search, this section presents the optimization methodology we are proposing.

Algorithm Interaction detection

Require: value of the decision variables $u = (\delta, \mathbf{l}, \mathbf{w})$ Initialize $\Phi_{tot} := 0$;

```

for i = 1 to N do                                     ▷ (for each trajectory i)
  Discretize the alternate trajectory i defined by  $u_i$  into a sequence  $\{P_{i,k}\}_{k=1}^{K_i}$ ;
  Initialize  $\Phi_i = 0$ ;
  for k = 1 to  $K_i$  do                                   ▷ (for each point  $P_{i,k}$  of trajectory i)
    Initialize  $\Phi_{i,k} := 0$ ;
    Compute the cell  $I_x, I_y, I_z, I_t$  corresponding to  $P_{i,k}$ ;
    Compute  $\Phi_{i,k}(u)$ :
    for  $i_x = I_x - 1$  to  $I_x + 1$  do
      for  $i_y = I_y - 1$  to  $I_y + 1$  do
        for  $i_z = I_z - 1$  to  $I_z + 1$  do
          for  $i_t = I_t - \frac{2t_s}{t_s}$  to  $I_t + \frac{2t_s}{t_s}$  do
            if  $\exists j \neq i$  such that  $j \in (i_x, i_y, i_z, i_t)$  then
               $L :=$  list of all trajectory sample point  $P_j$  in  $(i_x, i_y, i_z, i_t)$ ;
              for  $l = 1$  to  $\text{length}(L)$  do
                 $P := L(l)$ ;
                compute conflict probability,  $\mathcal{P}_C = \mathcal{P}_C(P_{i,k}, P)$  using (14.1);
                if  $\mathcal{P}_C = 0$  then
                   $\mathcal{P}_C := \text{interp}(\mathcal{P}_{\{i, k\}}, P)$ ;
                end if
                 $\Phi_{i,k} := \Phi_{i,k} + \mathcal{P}_C$ ;
              end if
            end for
          end if
        end for
      end for
    end for
     $\Phi_i := \Phi_i + \Phi_{i,k}$ ;
  end for
   $\Phi_{tot} := \Phi_{tot} + \Phi_i$ ;
end for
Return  $\Phi_{tot}$ .

```

Fig. 14.14 Interaction detection algorithm

14.5.1 Simulated Annealing: A Brief Overview

Simulated annealing is inspired by the annealing process in metallurgy where the state of material can be modified by controlling the cooling temperature. The physical annealing process consists in heating up a material to bring it to a high energy state. Then, it is slowly cooled down, keeping each given temperature stage for a sufficient duration until a thermodynamic balance is reached. The temperature is reduced according to a pre-described temperature reduction schedule, until the material reaches a global-minimum energy state and forms a crystallized solid. Decreasing too rapidly the temperature can however yield a non-desirable local minimum energy state.

In the simulated annealing optimization algorithm, the objective function to be minimized is analogical to the energy of the physical problem, while the decision variables of the problem correspond to the coordinates of the material's particles. A control parameter, T , that decreases as the number of iteration grows, plays the role of the temperature schedule, and a number of iterations, N_I , at each temperature step plays the role of time duration.

For a physical system, when the system reaches the thermodynamic balance at a given temperature, T , the energy, E , of its particles is distributed according to the *Boltzmann distribution*: $e^{\frac{-E}{k_B T}}$, where k_B is the Boltzmann constant. To simulate this evolution of the physical system towards the thermal equilibrium, the *Metropolis algorithm* is used. For a given temperature, T , starting from a current configuration, the state space of the simulated system is subjected to a transformation (e.g. apply a local change to one decision variable). If this transformation improves the objective function value, then it is accepted. Otherwise, it is accepted with a probability

$$P_{accept} := e^{\frac{\Delta E}{T}},$$

where ΔE is the degradation of the objective function value.

Once the number of iterations, N_I is reached, the temperature is decreased according to a pre-defined cooling schedule. As the temperature decreases, the probability, P_{accept} , to accept a degrading solution also decreases. The algorithm will eventually converge to a local optimum whose value is close to that of the global optimum if the temperature was decreased sufficiently slowly. The simulated annealing algorithm used here is summarized in Fig. 14.15, where S_c represents a current solution, and S_N represents a neighboring solution generated by a *neighborhood function*, which will be presented in the following section.

14.5.2 Iterative-Improvement Local Search: A Brief Overview

An iterative-improvement local search is an algorithm that starts from a given initial solution, and then iteratively replaces the current solution with a better solution chosen in a pre-defined neighborhood. Given an initial solution, the iterative-improvement local search generates a neighborhood solution, and then accepts this new solution only if it yields an improvement of the objective-function value. The algorithm stops when a (pre-defined) maximum number of iteration, N_{Loc} , is reached. The quality of the solution found by the local search depends on the initial solution, and on the definition of the neighborhood structure.

Let us denote by S_c the current solution, and let i be a given flight. In this work, we introduce a local-search module that relies on two different search strategies:

1. **Intensifying the search on one Particular Trajectory (PT).** This state-exploitation step focuses on improving S_c by applying a local change from a neighborhood structure involving solely flight i , so that only the decision variables $u_i = (\delta_i, l_i, w_i)$ can be modified.

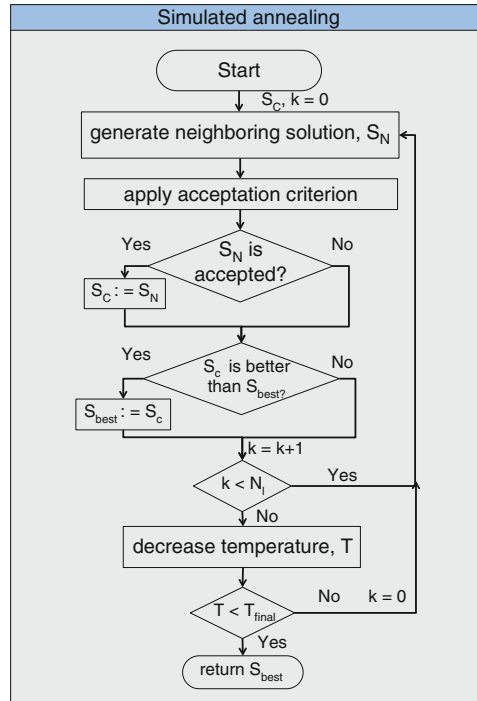


Fig. 14.15 Simulated annealing algorithm

2. **Intensifying the search on the Interacting Trajectories (IT).** This state-exploitation step applies a local change, from a neighborhood structure involving *several* flights: the flights that are currently interacting with flight i in solution S_c . Modifications to each of these *neighboring flights* are made sequentially in a greedy manner. For instance, suppose that trajectory i interacts with trajectories $p, q,$ and r . Changes are sequentially applied to the decision variables $u_p, u_q,$ and u_r . Obviously, the order in which the flights are considered may affect the quality of the resulting IT step.

14.5.3 Hybrid Simulated-Annealing/Iterative-Improvement Local Search

To implement the hybrid metaheuristic, we have to determine a structure to control the **level of hybridization** between each metaheuristic algorithm. According to [23], we may classify the level of hybridization into two levels:

- The *low-level hybridization* addresses an integration of metaheuristic algorithms, where each algorithm is strongly coupled with each other. In this case, an individual component in each metaheuristic may be replaced by, or exchanged, with

a component from another metaheuristic algorithm. For instance, one may consider using a greedy heuristic as a crossover operator in a genetic algorithm.

- For the *high-level hybridization*, each algorithm retains its own characteristics without direct interactions between the internal functions of the algorithm. Information are exchanged between each self-contained metaheuristic algorithm through a well-defined interface. For instance, the best solution obtained from one metaheuristic algorithm can be used as an initial solution for another metaheuristic algorithm.

Another property to consider is the **order of carrying out** each metaheuristic algorithm. In general, the algorithms can be run either in a sequential, an interleaved, or a parallel manners.

Finally, one has to determine the **values** of the various user-provided **parameters** specifying the metaheuristics. These parameters control, for example, the balance between exploration and exploitation of the solution space and must be fine-tuned with care as they have a strong impact on the quality of the solution obtained.

For the sake of simplicity in this preliminary implementation, the simulated annealing and the iterative-improvement local search are hybridized in a self-contained (high-level) manner where each algorithm is sequentially run. The iterative local search is integrated as an *inner loop* in the simulated annealing (SA) algorithm so that the local search is considered as one search step of the SA. The initial solution of the local search is provided by the current solution of the simulated annealing. The solution found by the local search is returned to the simulated annealing, where an acceptance condition will be systematically applied.

The order of carrying out each metaheuristic is given as follows.

- At each iteration of the hybrid algorithm, one flight is randomly chosen among all flights featuring a certain, pre-defined level of interaction. More precisely, let Φ_τ be a pre-defined interaction threshold value (provided by the user). The hybrid algorithm chooses randomly one flight in the set $\{i \in \{1, 2, \dots, N\} : \Phi_i \geq \Phi_\tau\}$. Let i denote the selected flight.
- Then, the hybrid algorithm determines whether to perform a classical SA step, or to trigger the iterative-improvement local search, or to perform both search strategies successively. This decision is taken according to a specific (user-defined) probability that depends upon the control temperature, T , and the value of the term, Φ_i of the objective function corresponding to flight i .

The **probability to carry out an SA** step, P_{SA} , is:

$$P_{SA}(T) = P_{SA,min} + (P_{SA,max} - P_{SA,min}) \cdot \frac{T_0 - T}{T_0}, \quad (14.8)$$

where $P_{SA,max}$ and $P_{SA,min}$ are the (user-provided) maximum and minimum allowed probabilities to perform SA (pre-defined by the user).

The **probability of running the iterative-improvement local search** module, P_{Loc} , is given by:

$$P_{Loc}(T) = P_{Loc,min} + (P_{Loc,max} - P_{Loc,min}) \cdot \frac{T_0 - T}{T_0}, \tag{14.9}$$

where, similarly, $P_{Loc,max}$ and $P_{Loc,min}$ are the (user-provided) maximum and minimum probabilities to perform the local search.

Finally, the **probability of carrying out both SA and the local search** (successively), P_{SL} , is simply:

$$P_{SL}(T) = 1 - (P_{SA}(T) + P_{Loc}(T)) \tag{14.10}$$

A key factor in tuning this hybrid algorithm is to reach a good trade-off between exploration (diversification) and exploitation (intensification) of the solution space, i.e. a compromise between fine convergence towards local minima, and the computation time invested in exploring the whole search space in order not to miss a global optimum.

The proposed hybrid algorithm is detailed in Fig. 14.16, where T_{init} and T_{final} are respectively the initial and the final temperatures of the (user-provided) cooling schedule, and where N_I is the maximal number of iterations at each temperature step (also set by the user).

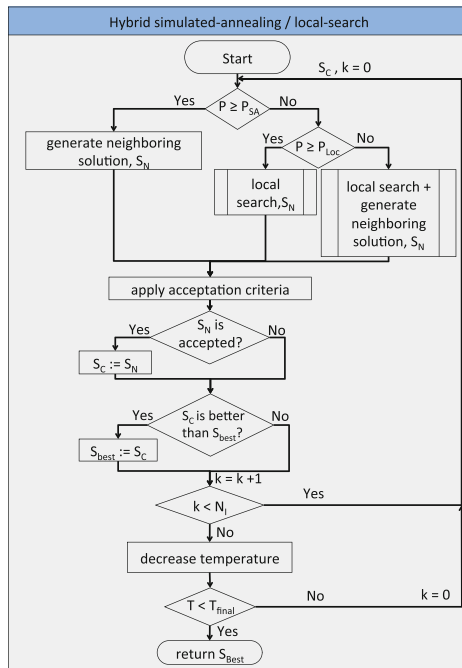


Fig. 14.16 Hybrid simulated-annealing/iterative-improvement local search algorithm

14.5.4 Neighborhood Function

A neighboring solution is generated by applying a so-called *neighborhood function* (or transformation operator) that generates a local change to the current solution. This change should be computed rapidly, but should not involve a drastic change in the current solution. Otherwise, the characteristics of the SA will become those of a pure random search.

To generate a neighboring solution, first a flight, i , to be modified is chosen. Then, one has to determine whether to modify the location of waypoints, or to modify its departure time, or its flight level. In general, searching for a neighboring solution in the time domain would be more preferable, since it does not induce extra fuel consumption. However, empirical tests show that limiting the search to only that degree of freedom results in prohibitive computational times before reaching a reasonably good solution.

Therefore, we introduce further user-defined parameters: $0 \leq P_w \leq 1$, $0 \leq P_{FL} \leq 1$, and P_δ to control respectively the probabilities of modifying: the location of the waypoints, the flight level, and the departure time of the chosen flight i . Once P_w and P_{FL} are chosen, P_δ is simply defined to satisfy $P_w + P_{FL} + P_\delta = 1$. These parameters allow the user to implement his preferences for the resulting conflict-free 4D trajectories.

14.6 Computational Results

The proposed strategic 4D trajectory planning methodology is implemented with the programming language Java on an AMD Opteron 2 GHz processor with 128 Gb RAM.

The methodology is tested on air traffic data representing a full-day en-route air traffic over the French airspace. It consists of $N = 8,836$ trajectories. Figures 14.17 and 14.18 illustrate the initial given trajectory sampling with $t_s = 60$ s in both the horizontal and the vertical planes (the dense area located at the coordinate point $(0; 5 \times 10^6)$ on Fig. 14.17 corresponds to Paris).

The parameter values that specify the problem under consideration are given in Table 14.1. Simply to give an idea of the complexity of the computation of the objective function of this problem instance; when using the sampling time-step value $t_s = 20$ s, the N trajectories are discretized into between 1,388,080 and 2,175,928 sample 4D points according to the location of waypoints used to modify the shape of trajectories. With regard to the dimension of the search space, remark that our optimization problem involves for this instance:

- $2MN = 53,016$ (continuous) waypoint variables (\mathbf{w});
- $N \frac{|\delta_a^i| + |\delta_d^i|}{\delta_s} + 1 = 3,189,796$ (discrete) departure-time shifts variables (δ);
- $N(2l_{i,max} + 1) = 44,180$ (discrete) flight-level shifts variables (\mathbf{I});

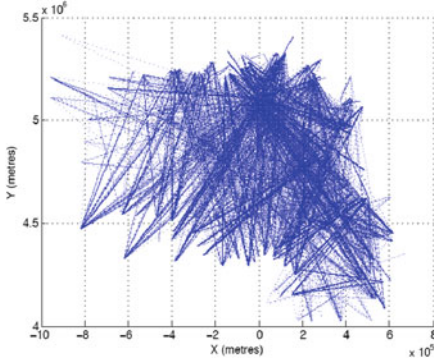


Fig. 14.17 The initial given trajectories consisting of a full-day traffic over the French airspace in the horizontal plane

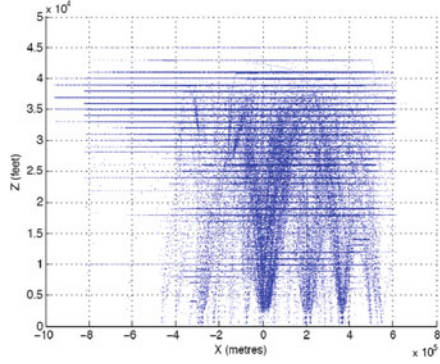


Fig. 14.18 The initial given trajectories consisting of a full-day traffic over the French airspace in the vertical plane

for a total of 53,016 continuous variables and 3,233,976 discrete variables. Empirical tests lead us to set the values of the parameters of the hybrid simulated-annealing/local-search as presented in Table 14.2.

Table 14.1 (User-defined) parameter values of the problem

Parameters	Value
Sampling time step, t_s	20 s
Inner-loop interpolation sampling time step, t_{interp}	5 s
Maximum departure time shift, $\delta_a^i = \delta_d^i$	60 min
Discretization time step for possible delay/advance departure-time shift	20 s
Maximum allowed route length extension, d_i	0.20
Maximum number of flight level shifts, $l_{i,max}$	2
Maximum number of waypoints, M	3
Probability to modify horizontal flight profile, P_w	1/3
Probability to modify flight level, P_{FL}	1/3
Probability to modify departure time, P_δ	1/3

Table 14.2 Empirically-set (user-defined) parameter values of the hybrid algorithm

Parameter	Value
Number of iterations at each temperature step, N_I	200
Number of iterations of the inner loop local search, N_{Loc}	5
Geometrical temperature reduction coefficient, β	0.99
Final temperature, T_{final}	$(1/500) \cdot T_{init}$
Probability to carry out simulated annealing, P_{SA}	$0.8 + 0.1 T/T_{init}$
Probability to carry out the inner-loop local search, P_{Loc}	$0.4 + 0.2 T/T_{init}$

The initial temperature, T_{init} , is calculated using an algorithm proposed in [13, pp. 44–45]. It is computed by initiating 100 deteriorating disturbances at random; evaluating the average variation ($\Delta\Phi_{avg}$) of the objective-function value; and then deducing T_{init} from the relation: $e^{-\frac{\Delta\Phi_{avg}}{T_{init}}} = \tau_0$, where τ_0 is the initial rate of accepting degrading solutions whose value depends on the assumed quality of the initial configuration. Empirical tests leads us to set $\tau_0 = 0.3$. Each temperature reduction is performed using a geometric reduction coefficient, β , whose value is provided by the user: $T_{k+1} := \beta T_k$.

The simulations are performed considering successively aircraft time uncertainty of 2 and 3 min, respectively. The proposed strategic trajectory planning methodology is able to find interaction-free trajectory plans for both cases. The required computation time for each problem is presented in Table 14.3. When considering higher level of time uncertainty (3 min), the solution space becomes more constrained and therefore the algorithm requires more computation time to converge.

The number of modified flight plans for both cases is compared in Fig. 14.19. The average changes made to the initial flight plan is shown in Table 14.4. The proposed algorithm is able to ensure separation of these nation-wide scale aircraft trajectories by modifying roughly 50% of the initial flight plans, yielding an average of 5% route length extension, 1.5 flight level shifts, and 30 min departure-time shifts.

Table 14.3 Numerical results considering aircraft time uncertainty of 2 and 3 min

Time uncertainty interval ($2t_\epsilon$) (min)	Initial Φ_{tot}	Resulting Φ_{tot}	Computation time (min)	No. of iterations performed
2	217,441.37	0.0	188.07	653,970
3	274,953.55	0.0	2210.42	5,583,192

Table 14.4 Average modifications applied to the initial flight plan

Time uncertainty interval ($2t_\epsilon$) (min)	Avg. route length extension ($\%L_{i,0}$)	Avg. FL shifts	Avg. departure- time shifts (min)
2	5.43	1.55	30.37
3	5.66	1.55	30.15

14.7 Conclusions

We introduced an efficient methodology to address the strategic trajectory planning problem in the framework of future trajectory-based ATM operations assuming time uncertainty on the position of the aircraft along its 4D trajectory. The proposed methodology minimizes interaction (a sum of conflict probabilities) between trajec-

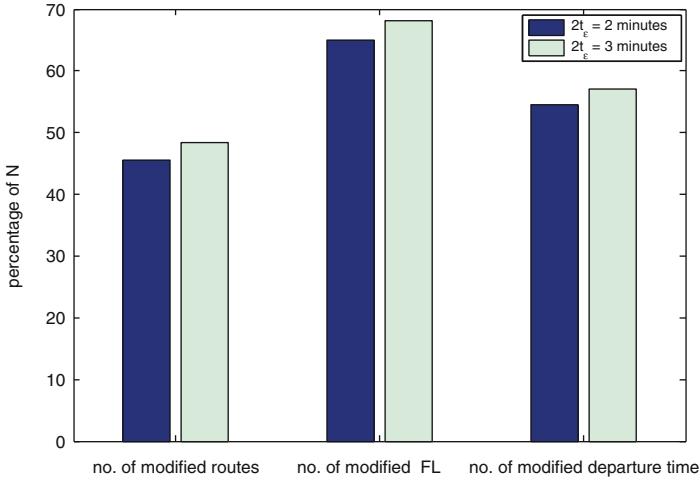


Fig. 14.19 Number of modified flight plans considering time uncertainty of 2 and 3 min

tories and results in conflict-free flight plans on a nation-wide scale data instance. Uncertainty of aircraft arrival time at a given position is taken into account by modeling the predicted aircraft arrival time as a triangular probability distribution over the uncertainty time period. The proposed trajectory planning approach relies on a route/flight-level/departure-time allocation technique to separate the aircraft trajectories. The problem was modeled mathematically under the form of a mixed-integer optimization problem aiming at minimizing total interaction between trajectories.

The problems we aim at addressing involve a full-day of traffic at the nation-wide scale involving more than 8000 trajectories. In order to ensure separation between trajectories under time uncertainty, we developed an efficient grid-based conflict detection module. To reduce the number of sampling points needed while minimizing further the computation time, this interaction-detection method interpolates the aircraft position between two suspected sampling points instead of refining the sampling-time step.

To find an optimal route, a flight level, and a departure time for each flight, we rely on a hybrid-metaheuristic optimization algorithm that combines the advantages of simulated annealing and of an iterative-improvement local-search method. The simulated annealing part ensures diversity of the candidate solutions considered, while the local-search module intensifies the search in promising regions of the feasible domain in order to accelerate convergence.

Computational experiments on a day instance of en-route air traffic over the French airspace with different levels of time uncertainty show that the proposed methodology is able to find interaction-free (i.e. conflict-free) trajectory plans, and to ensure separation between aircraft trajectories in presence of time uncertainty, within a computation time viable for the strategic planning level.

Further research should concentrate on reducing the extra route length, and the number of flight level shifts, and of departure-time shifts, instead of being content

with (possibly costly) interaction-free solutions. Another challenging research track is to take into account equity between airlines in the trajectory optimization process when deciding which trajectory is to be modified in order to avoid a conflict.

Acknowledgements This work has been supported by French National Research Agency (ANR) through JCJC program (project ATOMIC n° ANR 12-JS02-009-01).

References

1. A. Agustín, A. Alonso-Ayuso, L.F. Escudero, C. Pizarro, Mathematical optimization models for air traffic flow management: a review, in *Combinatorial Optimization in Practice*, ed. by A. Bui, I. Tseveendorkj. Studia Informatica Universalis, vol. 8 (Hermann Informatique, Paris, 2010), pp. 141–184
2. A. Akgunduz, B. Jaumard, G. Moeini, Non-time indexed modeling for en-route flight planning with speed-fuel consumption trade-off, in *2nd International Conference on Interdisciplinary Science for Innovative Air Traffic Management*, Toulouse, 2013
3. N. Barnier, C. Allignol, 4D - trajectory deconfliction through departure time adjustment, in *8th USA/Europe Air Traffic Management Research and Development Seminar (ATM 2009)*, Napa, CA, 2009
4. N. Barnier, C. Allignol, Combining flight level allocation with ground holding to optimize 4D-deconfliction, in *USA/Europe ATM Seminar: Air Traffic Management Research and Development Seminar*, Berlin, 2011
5. D. Bertsimas, S. Patterson, The air traffic flow management problem with en-route capacities. *Oper. Res.* **46**(3), 406–422 (1998)
6. D. Bertsimas, G. Lulli, A. Odoni, The air traffic flow management problem: an integer optimization approach, in *Integer Programming and Combinatorial Optimization*. Lecture Notes in Computer Science, vol. 5035 (Springer, Berlin, 2008), pp. 34–46
7. D. Bertsimas, G. Lulli, A. Odoni, An integer optimization approach to large-scale air traffic flow management. *Oper. Res.* **59**(2), 211–227 (2011)
8. S. Cafieri, N. Durand, Aircraft deconfliction with speed regulation: new models from mixed-integer optimization. *J. Glob. Optim.* **54**(4), 613–629 (2013)
9. S. Chaimatanan, D. Delahaye, M. Mongeau, A methodology for strategic planning of aircraft trajectories using simulated annealing, in *International Conference on Interdisciplinary Science for Innovative Air Traffic Management*, Daytona Beach, FL, 2012
10. S. Chaimatanan, D. Delahaye, M. Mongeau, Strategic deconfliction of aircraft trajectories, in *2nd International Conference on Interdisciplinary Science for Innovative Air Traffic Management*, Toulouse, 2013
11. S. Constans, B. Fontaine, R. Fondacci, Minimizing potential conflict quantity with speed control, in *4th Eurocontrol Innovative Research Workshop and Exhibition*, Bretigny-sur-Orge, 2005, pp. 265–274
12. D. Delahaye, S. Puechmorel, *Modeling and Optimization of Air Traffic* (Wiley-ISTE, North America, 2013)
13. J. Dreco, A. Petrowski, P. Siarry, E. Taillard, *Metaheuristics for Hard Optimization* (Springer, Berlin, 2006)
14. N. Durand, J.B. Gotteland, Genetic algorithms applied to air traffic management, in *Metaheuristics for Hard Optimization* (Springer, Berlin, 2006), pp. 277–306
15. D. Gianazza, N. Durand, Separating air traffic flows by allocating 3D-trajectories, in *DASC 04. The 23rd Digital Avionics Systems Conference, 2004*, vol. 1 (2004), p. 2.D.4-21-13
16. I. Hwang, C.E. Seah, Intent-based probabilistic conflict detection for the next generation air transportation system. *Proc. IEEE* **96**(12), 2040–2059 (2008)

17. J.K. Kuchar, L.C. Yang, A review of conflict detection and resolution modeling methods. *IEEE Trans. Intell. Transp. Syst.* **1**, 179–189 (2000)
18. J. Lohn, J. Rios, A comparison of optimization approaches for nationwide traffic flow management, in *AIAA Guidance, Navigation and Control Conference*, Chicago, 2009
19. A.R. Odoni, The flow management problem in air traffic control, in *Flow Control of Congested Networks*, ed. by A.R. Odoni, L. Bianco, G. Szego (Springer, Berlin, 1987), pp. 269–288
20. S. Oussedik, Application de l'Évolution artificielle aux problèmes de congestion du trafic aérien. PhD thesis, École Polytechnique, Palaiseau, 2000
21. S. Oussedik, D. Delahaye, Reduction of air traffic congestion by genetic algorithms, in *PSSN V: Parallel Problem Solving from Nature*. Lecture Notes in Computer Science, vol. 1498 (Springer, Berlin, 1998), pp. 855–864
22. M. Prandini, J. Hu, J. Lygeros, S. Sastry, A probabilistic approach to aircraft conflict detection. *IEEE Trans. Intell. Transp. Syst.* **1**(4), 199–220 (2000)
23. E-G. Talbi, *Metaheuristics: From Design to Implementation* (Wiley, Hoboken, 2009)
24. M. Terrab, A.R. Odoni, Strategic flow management for air traffic control. *Oper. Res.* **41**(1), 138–152 (1993)

Chapter 15

Sampling-Based Genetic Algorithms for the Bi-Objective Stochastic Covering Tour Problem

Michaela Zehetner and Walter J. Gutjahr

Abstract The paper investigates a sampling-based extension of the NSGA-II algorithm, applied to the solution of a bi-objective stochastic covering tour problem. The proposed extension uses variable samples for gradually improving approximations to the Pareto front. The approach is evaluated on a test benchmark for a humanitarian logistics application with data from Senegal. Comparisons to alternative solution techniques, in particular also to the exact solution of the deterministic counterpart problem based on a fixed sample, show the superiority of our approach.

Keywords Humanitarian logistics • Genetic algorithms • Multi-objective optimization • Stochastic optimization • Covering tour problem

15.1 Introduction

“On a Saturday morning in March 2003, I got a call from our regional management team in Amman requesting an urgent airlift of emergency supplies, materials and vehicles. I immediately called our head logistician who proceeded to make calls to our logistics staff in Italy, Germany and the US. By Monday morning bids were being answered. By Tuesday morning the transporter had been selected and mobilised. By Wednesday morning all the goods were prepared for shipment. By Thursday morning the aircraft was on the tarmac at Brindisi airport. That afternoon it landed with 40 tonnes of goods in Amman and was cleared and offloaded within a couple of hours. Three transport trucks, 10,000 collapsible water containers and purification tablets, 6,300 blankets and 1,800 plastic tarpaulins were among the goods landed. By the weekend – seven days after the initial phone call – these goods were en route to regional destinations in preparedness and readiness for possible influx of refugees from Iraq.” [16].

When disasters occur – like in the example above – a good logistics management is very critical. The affected people are to be provided with necessities like food, water, medicines and sanitation as quickly as possible. The big challenge is to get

M. Zehetner • W.J. Gutjahr (✉)
University of Vienna, Wien, Austria
e-mail: walter.gutjahr@univie.ac.at

these things to the right place at the right time and at the right cost. Decisions have to be made for example about procurement, transportation, warehousing, inventory management and others. These are some of the important tasks of humanitarian logistics (cf. [50]). Generally, the field of humanitarian logistics is growing and also has to because the number of disaster relief operations will possibly increase fivefold within the next 50 years (cf. [49]).

In Tricoire et al. [51], a *bi-objective stochastic covering tour problem* concerned with disaster management is formulated and solved using the framework of the epsilon-constraint algorithm. One objective of the problem is to minimize the expected uncovered demand of the affected people, and the other objective is to minimize total costs. Total costs include the cost for opening distribution centers and the cost for carrying out the delivery tours. Because of possible short-term environmental changes and other uncertainty factors, demand cannot be assumed as deterministically known and is therefore modeled by a probability distribution. A branch-and-cut solution algorithm is used for determining the set of Pareto-optimal solutions, based on a comparably small set of randomly chosen demand scenarios. In view of the inherent complexity of the problem, however, it is not possible to solve medium-sized or large instances to optimality.

In this paper, another way to address this problem will be studied. Whenever the distribution defined by a set of scenarios is rather an approximation to a true stochastic model than a proper description of this model itself, the solution of the scenario-based optimization problem cannot be considered as the *exact* solution of the original problem anyway. Thus, the solution quality with respect to the original problem may possibly even be increased by solving the underlying deterministic problem by a powerful *heuristic* instead of a mathematical programming algorithm, and by using the saved computation time for including a much higher number of scenarios.¹ Simultaneously, this also allows to deal with larger problem instances.

For solving the underlying (in our case multi-objective) deterministic problem heuristically, we have chosen the well-established NSGA-II metaheuristic, developed by Deb et al. [8]. Clearly, our multi-objective application problem is special insofar as it builds on a stochastic model, which means that objective function values can only be estimated by sampling. This raises the question of a suitable collaboration between the NSGA-II algorithm and a sampling procedure. As we shall see, this collaboration can be organized in very straightforward ways, which will provide us with two basic approaches. As a third alternative, however, also a more sophisticated interplay may be used: We shall apply the Adaptive Pareto Sampling (APS) framework proposed in [19] for the solution of a broad class of stochastic multi-objective optimization problems. This approach has appealing theoretical features: On mild conditions, convergence to the true Pareto front can be ensured [19, 23], and it can be shown analytically that the optimization time overhead compared to the solution of the corresponding *deterministic* multi-objective counterpart problem

¹ It should be mentioned that also in these cases, the exact solution of the sample-average approximation problem has its own advantages, e.g., it allows the construction of confidence intervals on the optimality gap (cf. [37]).

is only moderate [21]. Of course, the performance in an application to a given problem type has to be evaluated experimentally. The present chapter contains results of this kind.

The chapter is organized as follows. Section 15.2 presents the problem to be investigated, first in informal, then in mathematical terms. In Sect. 15.3, the solution approaches to be compared and the used algorithms (NSGA-II and APS) are described. Section 15.4 provides the experimental results for a real-world data benchmark. In the last section, conclusions from our results are summarized.

15.2 The Bi-Objective Stochastic Covering Tour Problem

In this section, we start by recapitulating the covering tour problem (CTP) and explain then the specific extended variant (first introduced in [51]) of the CTP investigated in this paper in informal terms, as it occurs in the particular application context considered here. Furthermore, a short literature review is given referring to theoretical approaches as well as practical applications regarding the CTP and some of its variants. Then, the extended problem to be investigated in this paper is described on a formal mathematical level. Because of the solution approach adopted in this paper, it is also necessary to briefly recall the vehicle routing problem, which occurs as a subproblem and therefore needs to be solved as well.

15.2.1 The Covering Tour Problem

The CTP is a combinatorial optimization problem combining features of facility location and vehicle routing. Its basic version is described in Gendreau et al. [15]. Therein, the problem is defined on an undirected graph $G = (V \cup W, E)$. The vertices in the set V can be visited, whereas the vertices in the set $T \subseteq V$ must be visited. Vertex $v_1 \in T$ denotes the depot. All vertices in W must be covered. A vertex $w \in W$ is called covered if the distance between w and the nearest visited vertex $v \in V$ is less than or equal to the predefined covering distance c . The set $E = \{(v_i, v_j) \mid v_i, v_j \in V \cup W, i < j\}$ is the edge set. The distance between two vertices is stored in the distance matrix $D = (d_{ij})$. The objective function is the minimization of the tour length under the given coverage constraints. Note that for $c = 0$, the CTP reduces to a traveling salesperson problem (TSP).

The CTP and its variants can be applied, e.g., in the areas of transportation network design, location routing and telecommunication, and they are also suitable to address various other problems existing in reality.

15.2.2 The Bi-Objective Stochastic Extension

Variants of the CTP can be used to address decision problems occurring (among others) in the humanitarian logistics sector. Let us consider the case of a disaster

relief operation. In the aftermath of a disaster, like a flood or an earthquake, the affected people are to be provided with essential goods like water, food or medicine as quickly as possible. To supply these people with relief goods, vehicle routes for transportation have to be determined. The vehicles should deliver the goods from a depot to certain distribution centers. People whose villages cannot be supplied directly will have to walk to the nearest distribution center. The maximum distance they are able to traverse corresponds to the covering distance in the context of the CTP. It is assumed that a distribution center can only be located in a village, i.e., the set of potential locations is equal to the set of villages. The distances between all pairs of the villages are assumed to be given.

The problem proposed in [51] takes an additional feature into account in order to better adapt the model to reality: Immediately after the occurrence of a disaster, it might not always be clear to what extent the affected people will need relief goods during the period of the disaster relief operation, which can take some weeks or in some cases even months. Nevertheless, it is important to establish fixed, reliable delivery schemes and to organize the procurement of the goods from national and international markets in time. Therefore, a decision problem under uncertainty arises. To take account of this aspect, the model represents the demand in each village as a random variable, assuming that the joint distribution of these random variables can be estimated at the time of the decision, but not their actual realizations. This produces a *stochastic* optimization problem. Evidently, by each decision on the organization of the delivery (except a very conservative one, which would be too costly in cases of high uncertainty), it can happen that some part of the demand remains uncovered. Of course this is very undesirable, so the (expected) amount of uncovered demand should be minimized.

Distribution centers are assumed to have only limited capacities to store or provide relief goods. This is not only because disasters usually happen unexpectedly and therefore only improvised distribution centers are at hand, but also because in order to ensure a controlled delivery of the goods and to prevent plundering, staff has to be assigned to each distribution center, which further restricts the throughput in addition to storage limitations. If a potential distribution center is selected to be opened, costs will arise which may depend on the specific location. A given fleet of vehicles provides the distribution centers with the goods to be delivered there. Each vehicle starts its tour at a depot and returns there.

Another extension of the CTP proposed in [51] concerns coverage. The extended model diverges from the assumption that a population center (a village) can only either be covered completely or not at all. A step function is used to describe the percentage of inhabitants of a village who are able or willing to go to the closest distribution center if it is located at a certain distance d . The percentage of people who can be covered in this way decreases with increasing value of d .

Finally, the problem is formulated as a *bi-objective* (stochastic) optimization problem. The first objective is to minimize all costs including opening costs and routing costs. The second objective is to minimize expected uncovered demand. These goals can be seen as conflicting. To illustrate this, let us consider two extreme cases. Firstly, let us assume that we reduce costs to zero. This would imply that no distribution centers are opened and nobody can be supplied with any relief goods,

which results in the highest possible amount of uncovered demand. Alternatively, if we implement a solution ensuring that the demand of all people is satisfied (in the extreme case, this would imply to establish a distribution center in every village), the expected costs will be the highest. To take account of the trade-off between the monetary and the non-monetary objective, [51] uses the concept of Pareto optimality. Decision makers will be provided with various feasible solutions that are all guaranteed to be efficient in the Pareto sense. Out of these solutions, the decision makers can choose one which fits their preferences best.

In this paper, we will retain the problem formulation of Tricoire et al. [51] and even use the same test benchmark, but apply different solution techniques. The contribution of the present paper is twofold. First, we shall describe heuristic approaches to solve the problem outlined above. Secondly, we will study possible options concerning the sampling part of the algorithms by examining whether the use of a more sophisticated sampling technique can improve the quality of the solutions under fixed given runtime. Of course, it is also interesting to know whether the mathematical programming approach (based on a limited sample size) or the heuristic approach (which can use a much larger sample size) will produce better results within a comparable computation time, if the results of both are assessed on a very large sample.

15.2.3 Literature Review

Current and Schilling [7] introduced the maximal covering tour problem (MCTP), where exactly p out of n nodes have to be visited on a tour. One objective is to minimize the total tour length and the second is to maximize the total demand which can be covered. Coverage means that a demand point lies within a pre-determined maximum distance from a node which is on the tour. Gendreau et al. [15], as mentioned above, formulate a single-objective version of the CTP. They develop a first exact method, namely a branch-and-cut algorithm and also present a heuristic to solve this problem. In Hachicha et al. [25], three heuristics are proposed for the multi-vehicle CTP (m -CTP) with m identical vehicles, where the goal is to minimize the total length of all vehicle routes. Motta et al. [39] introduce three GRASP metaheuristics for solving a generalized version of the CTP. Baldacci et al. [3] tackle the CTP by using three heuristic scatter search algorithms.

In Jozefowicz et al. [33], a bi-objective covering tour problem is studied, where the first objective is the minimization of the tour length and the second objective is to minimize the largest distance between a node in a set W that has to be covered and the respective nearest node lying on the tour. A combination of a multi-objective evolutionary algorithm and a branch-and-cut algorithm is used to determine the optimal solutions. Naji-Azimi et al. [40] analyze an extension of the m -CTP, where multiple commodities and a fleet of heterogeneous and capacitated vehicles are assumed and split deliveries are allowed. To solve the problem, they propose an efficient heuristic approach and test it on real data. Jozefowicz [31] presents a col-

umn generation approach and a branch-and-price algorithm for the m -CTP [32]. Ha et al. [24] develop a metaheuristic and an exact branch-and-cut algorithm to solve a special case of the m -CTP, where the restriction concerning the length of each route is relaxed. It turns out that the exact method performs better than the column generation approach of Jozefowicz [31]. Nolz et al. [41] are concerned with the distribution of drinking water in a post-natural-disaster situation. In Nolz et al. [42], a multi-objective CTP is formulated for the problem of delivering relief supplies after the occurrence of a disaster. The three objective functions incorporate a measure of risk, a combination of two coverage criteria and a measure of total travel time. In Rath and Gutjahr [43], a multi-objective model is assumed where contrary to the above-mentioned papers, relief goods are delivered from the distribution centers to the villages by vehicles. A two-stage stochastic programming extension of this model is investigated in Rath et al. [44].

The CTP and its variants have also been applied to real-world problems in the area of medical supply. Earlier studies already focus on mobile health care services in developing countries (see Foord [14] and Swaddiwudhipong et al. [46]). Hodgson et al. [26] make use of the CTP model to determine a minimum length tour for one mobile health care facility in Suhum district in Ghana. They also consider various road types and different weather conditions in their model, to be better in line with reality. Doerner et al. [10] provide an extension of the model in Hodgson et al. [26] and Hachicha et al. [25]. They formulate a three-objective CTP, where the first criterion is economic efficiency, which is closely related to the tour length, the second is an average accessibility criterion and the third is related to coverage.

Literature on the solution methodology will be indicated in Sect. 15.3.

15.2.4 Mathematical Formulation

The problem formulation in this section is based on [51]. The problem under investigation can be defined on a complete undirected graph $G = (V_0, E)$, where $V_0 = V \cup \{0\}$ is the node set and E is the set of edges. Node 0 is the depot. The set V contains all population nodes and hence, by assumption, also all potential distribution centers. A population node is henceforth always referred to as a “village”. Although there is no need to differentiate between villages and potential distribution centers, index i defines a node as a village and index j defines it as a potential distribution center. The edges in $E = \{(i, j) \mid i \in V_0, j \in V_0, i < j\}$ are undirected. Therefore, the distance matrix D storing the distances d_{ij} between two nodes is symmetric. Driving costs depend on the distance between two nodes and are assumed as $\tau \cdot d_{ij}$ with τ denoting the cost of driving one kilometer.

We make the assumption that the people living in the villages are to be provided with a single commodity. (The case of multiple commodities can be addressed by the same model, as long as divisibility problems are not relevant and the ratios between the required amounts of the single commodities are constant.) The vehicles $k \in K$ can be used to transport goods to the distribution centers, but vehicle k is limited to the capacity Q_k ($k \in K$). The cost of opening a distribution center in node j is c_j

currency units. A distribution center at location j has a capacity of γ_j ; this value can vary across locations. The demand in village $i \in V$ is given by the random variable W_i . The joint distribution of the random variables W_i ($i \in V$) is assumed to be known. More precisely, it is assumed that $W_i = \xi_i \cdot w_i$, where the ξ_i are random variables which take account of uncertainty, and $w_i = \mathbf{E}(W_i)$ denotes the expected value of W_i . The values w_i are essentially determined by the population sizes (number of inhabitants) of the villages i . We assume that the inhabitants of a village always envisage the nearest distribution center j as a possible source for obtaining supply. However, the percentage of people who are able (or willing) to go to this nearest distribution center j is determined by a decreasing function ψ of the distance d_{ij} . This means that the farther they would have to walk, the fewer people will walk and consequently receive goods.

The first decision in this problem concerns the question where to open distribution centers. Depending on this choice, tours of the available fleet of vehicles have to be determined to visit the opened distribution centers and to provide them with goods. We refer to this part of the problem as the *first-stage decision*; it is made before the actual demand is known. Note that we also include the choice of the tours in the first-stage decision, which means that the tours are not changed anymore at a later time. In the disaster relief application, this is desired insofar as it gives each driver a responsibility for a certain area s/he has to supply with the relief items. In the *second-stage decision*, the quantities of goods actually delivered to each distribution center on each tour have to be determined, depending on the current demand, the capacity of the distribution center and the load capacity of the vehicle.

The decision variables used in the model are the following:

- x_{ijk} counts how often vehicle k passes through edge (i, j) ,
- $y_{ij} = 1$ means that the distribution center in node j is the closest to village i , otherwise $y_{ij} = 0$,
- $z_{jk} = 1$ means that vehicle k visits the distribution center in node j , otherwise $z_{jk} = 0$,
- u_{jk} defines the quantity delivered by vehicle k to node j .

We use x , y , z and u for the respective matrices or arrays of decision variables. Moreover, $\delta(S) = \{(i, j) \in E \mid i \in S, j \in V_0 \setminus S \text{ or } j \in S, i \in V_0 \setminus S\}$ contains all edges for which exactly one incident node is in the set $S \subseteq V_0$. For a set $E' \subseteq E$, we use the abbreviation $x_k(E')$ for denoting $\sum_{(i,j) \in E'} x_{ijk}$.

The bi-objective stochastic CTP outlined before in informal terms can now be expressed by the following bi-objective two-stage stochastic program:

First stage:

$$\min_{x,y,z} (f_1, f_2) \quad \text{s. t.} \tag{15.1}$$

$$f_1 = \tau \sum_{k \in K} \sum_{(i,j) \in E} d_{ij} x_{ijk} + \sum_{k \in K} \sum_{j \in V} c_j z_{jk} \tag{15.2}$$

$$f_2 = \mathbf{E}(R(y, z, \xi)) \quad (15.3)$$

$$\sum_{j \in V} y_{ij} = 1 \quad \forall i \in V \quad (15.4)$$

$$y_{ij} \leq \sum_{k \in K} z_{jk} \quad \forall i, j \in V \quad (15.5)$$

$$\sum_{j \in V} d_{ij} y_{ij} \leq d_{im} + \mathcal{M} \left(1 - \sum_{k \in K} z_{mk} \right) \quad \forall i, m \in V \quad (15.6)$$

$$\sum_{k \in K} z_{jk} \leq 1 \quad \forall j \in V \quad (15.7)$$

$$\sum_{k \in K} z_{0k} = |K| \quad (15.8)$$

$$x_k(\delta(j)) = 2z_{jk} \quad \forall j \in V_0, k \in K \quad (15.9)$$

$$x_k(\delta(S)) \geq 2z_{jk} \quad \forall S \subseteq V, j \in S, k \in K \quad (15.10)$$

$$x_{ijk} \in \{0, 1\} \quad \forall (i, j) \in E \setminus \delta(0), \quad x_{ijk} \in \{0, 1, 2\} \quad \forall (i, j) \in \delta(0) \quad (15.11)$$

$$y_{ij} \in \{0, 1\} \quad \forall (i, j) \in V \quad (15.12)$$

$$z_{jk} \in \{0, 1\} \quad \forall j \in V_0, k \in K \quad (15.13)$$

Second stage:

$$R(y, z, \xi) = \min_u \left[\sum_{i \in V} \xi_i w_i - \sum_{k \in K} \sum_{j \in V} u_{jk} \right] \quad \text{s. t.} \quad (15.14)$$

$$u_{jk} \leq \sum_{i \in V} \xi_i w_i \psi(d_{ij}) y_{ij} \quad \forall j \in V, k \in K \quad (15.15)$$

$$u_{jk} \leq \gamma_j z_{jk} \quad \forall j \in V, k \in K \quad (15.16)$$

$$\sum_{j \in V} u_{jk} \leq Q_k \quad \forall k \in K \quad (15.17)$$

$$u_{jk} \geq 0 \quad \forall j \in V, k \in K \quad (15.18)$$

According to Eq. (15.1), the two objectives f_1 and f_2 of the first-stage problem should be minimized. The objective function f_1 [see Eq. (15.2)] includes the driving costs as well as the opening costs of the distribution centers. It is important to note that if $\sum_{k \in K} z_{jk} = 1$, a distribution center in village j is opened. Equation (15.3) defines f_2 as the expected uncovered demand, which arises either because the way to the nearest distribution center is too long for the inhabitants or due to capacity constraints of the distribution centers or the vehicles. The actual uncovered demand will be calculated in the second-stage problem.

Constraints (15.4) ensure that each village is assigned to exactly one distribution center. Furthermore, in this case the distribution center has to be an open one [see constraints (15.5)]. According to constraints (15.6), the people of each village go to the nearest opened distribution center. Here, \mathcal{M} is a very large number. Constraints (15.7) imply that there is not more than one vehicle allowed to visit a distribution center. These constraints also say that in a village there cannot be more than one opened distribution center. Because of constraint (15.8), all vehicles have to visit the depot. Constraints (15.9) contain degree constraints. Constraints (15.10) exclude infeasible sub-tours. Constraints (15.11) allow a vehicle to use the link between two villages at most once. On the other hand, the link between a village and the depot can be used at most twice (which allows forth-and-back tours where a vehicle just visits one village and then returns to the depot again). Constraints (15.12) and (15.13) state that y_{ij} and z_{jk} are binary variables.

The objective function (15.14) in the second-stage problem is defined as total demand minus total supply or, in other words, as the uncovered demand, which is to be minimized. According to constraints (15.15), the supply u_{jk} has to be less than or equal to the total request occurring in distribution center j . This request includes the requests from all villages that are assigned to distribution center j . Therein, the request from village i is calculated as the product of the overall demand $\xi_i w_i$ occurring in village i and the share $\psi(d_{ij})$ of people going to the distribution center in j to satisfy their demand. Because of constraints (15.16), the supply in a distribution center has also to be less than or equal to the capacity of this distribution center. Constraints (15.17) pay attention to the load capacity of each vehicle. Constraints (15.18) state that the supplies u_{jk} can only attain a value larger than or equal to zero.

Obviously, as soon as the subset of candidate sites where distribution centers are to be opened has been selected, a *vehicle routing problem* (VRP) remains to be solved as a subproblem: Each of the opened distribution centers has to be visited by exactly one of the available vehicles. Split deliveries are not allowed. The aim of the subproblem is to find routes for the vehicles at minimum total cost.

15.3 Algorithms

In this section, we will present the algorithms we use for solving the problem described above. Let us start with a short literature review on solution methods for stochastic multi-objective optimization problems (SMOOPs) in the sense of the de-

termination of Pareto-optimal solutions with respect to expected objective function values.² For more detailed information, the reader is referred to the surveys by Ben Abdelaziz [5], Gutjahr [20] and Gutjahr and Pichler [22]. We focus here on metaheuristic solution methods.

Most metaheuristics for SMOOPs use diverse variants of evolutionary algorithms. Here are a few examples: Hughes [28], Teich [48], and Eskandari et al. [13] adapt the multi-objective evolutionary algorithm SPEA and the non-dominated sorting genetic algorithm NSGA-II to the stochastic situation. Also Ding et al. [9] propose a multi-objective genetic algorithm and combine it with a simulation procedure. Gutjahr [18] uses multi-objective variants of Ant Colony Optimization and of Simulated Annealing and generalizes them to the stochastic case. Amodeo et al. [1] combine a discrete-event simulation procedure with the multi-objective algorithms SPEA-II and NSGA-II, and compare with a multi-objective variant of the Particle Swarm Optimization algorithm. Eskandari and Geiger [12] carry out computational experiments with an extension of the approach in [28]. Syberfeldt et al. [47] use a multi-objective evolutionary algorithm supported by an artificial neural network and combine it with simulation. Basseur and Zitzler [4] and Liefvooghe et al. [35, 36] reduce the multi-objective problem to a single-objective one by means of the *quality indicator* technique. Finally, the APS algorithm [19, 23] allows it to extend an arbitrary (exact or heuristic) multi-objective optimization algorithm to the stochastic optimization case by interleaving it with a sampling procedure. We shall describe APS in more detail below.

15.3.1 NSGA-II

Deb et al. [8] introduced the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). In genetic algorithms (GAs), solutions are encoded as individuals, also termed *chromosomes* $\mathbf{x} \in \mathbf{X}$; together, the individuals form a *population*. In iterative steps, new populations are generated by selecting parent chromosomes and creating offspring, using genetic operators like *crossover* and *mutation*. By repeating this procedure, the individuals in the population are expected to become fitter and fitter with respect to the objective function(s) under consideration.

In a multi-objective GA, the aim is to compute an approximation to the *Pareto front*. The Pareto front is the image of the set of all Pareto-optimal solutions in the objective space. A solution \mathbf{x} is called *Pareto-optimal* (or *efficient*) if it is not dominated by any other solution \mathbf{y} , where \mathbf{y} dominates \mathbf{x} if and only if $f_i(\mathbf{y}) \leq f_i(\mathbf{x})$ for all i and $f_i(\mathbf{y}) < f_i(\mathbf{x})$ for at least one i . (Minimization of all objective functions f_i is assumed.) We write $\mathbf{y} \prec \mathbf{x}$ if \mathbf{y} dominates \mathbf{x} (cf. [34]).

² We assume a risk-neutral decision maker, which implies that only the expectations of the objective functions count. In the case of a risk-averse decision maker, more complex methods are required, see [22]. Also the case of chance constraints will not be discussed here.

Algorithm 1 NSGA-II**Procedure** NSGA-II

generate P_0 of size \mathcal{N}	create initial population randomly
$\mathcal{F} = \text{non-dominated-sort}(P_0)$	$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \dots)$
crowding-distance-assignment(\mathcal{F}_i)	calculation of crowding distances
for ($t = 0$, break condition) {	
selection(P_t)	selection of parents
generate offspring Q_t of size \mathcal{N}	use crossover and mutation
$R_t = P_t \cup Q_t$	combination: parents and offspring
$\mathcal{F} = \text{non-dominated-sort}(R_t)$	
$P_{t+1} = \emptyset, i = 1$	
while $ P_{t+1} + \mathcal{F}_i \leq \mathcal{N}$ {	until population is filled
crowding-distance-assignment(\mathcal{F}_i)	
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i$	add front to population
$i = i + 1$	
}	
sort(\mathcal{F}_i) on crowding distance	in descending order
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (\mathcal{N} - P_{t+1})]$	include first $(\mathcal{N} - P_{t+1})$ elements
$t = t + 1$	
}	
return(\mathcal{F}_1)	return solutions from the first front

15.3.1.1 Main Procedure

A pseudo-code description of the NSGA-II is given above (Algorithm 1). The algorithm starts by randomly creating an initial population P_0 of size \mathcal{N} . These solutions are sorted and partitioned into *non-dominated fronts*. Then a so-called *crowding distance* is calculated for each solution. Chromosomes are selected according to their fitness to become parents of the child population. Crossover and mutation are used to create the offspring population Q_t of size \mathcal{N} . Then Q_t and P_t are combined to a population R_t of size $2\mathcal{N}$. Because the next aim is to achieve a new population P_{t+1} with a reduced population size \mathcal{N} instead of $2\mathcal{N}$, a non-dominated sorting procedure is applied to the population R_t . The fronts are included successively in the new population P_{t+1} , starting with the first front and so on. The fronts are inserted until there is no more space for the members of an additional front. If the new population P_{t+1} is not completely filled, the individuals of the current front that cannot be included completely anymore are sorted in descending order based on crowding distance. Then the first $\mathcal{N} - |P_{t+1}|$ solutions are included in P_{t+1} . Now the loop starts again by selecting parents to create offspring and so on. As soon as a termination condition is met, the algorithm stops and returns the non-dominated individuals belonging to the first front.

Jensen [30] proposed an efficient algorithm for *non-dominated sorting*, which reduces the overall computational complexity of the NSGA-II from $O(M\mathcal{N}^2)$ to $O(\mathcal{N} \log^{M-1} \mathcal{N})$, where M is the number of objectives. All the fronts are created simultaneously. The algorithm proceeds as follows. First, the solutions are all sorted in ascending order with respect to the first objective function value. If these values are the same, then the corresponding solutions are sorted with respect to the second objective function value. So for $i < j$, it always holds that $x_j \not\prec x_i$. Next, solution x_1 is assigned to the first front. The current number of fronts is stored in e . Now the iteration starts for $i = 2$ to \mathcal{N} . We examine whether any solution of the current worst front \mathcal{F}_e dominates the solution x_i . If not, then x_i belongs to one of the fronts created so far. But if the solution x_i is dominated by a solution belonging to \mathcal{F}_e , a new front is created and e is increased by one. Because of the pre-sorting and the non-domination-property in each front it is quite easy to check whether x_i is dominated by \mathcal{F}_j . Only a comparison between the value of the second objective of solution x_i and the value of the second objective of the last solution in \mathcal{F}_j is needed. The algorithm returns all fronts $\mathcal{F}_1, \dots, \mathcal{F}_e$.

Besides the aim of convergence to the Pareto front, a good spread of solutions is also desirable. To maintain diversity, Deb et al. [8] presented a so-called crowded-comparison approach. A crowding distance, which is a measure of how far solutions belonging to the same front are away from each other in the objective space, is computed for each solution. We implemented the crowding distance computation in a quite standard way, so we omit a detailed description.

The population members are compared with each other by using the *crowded-comparison operator*: If two solutions belong to the same non-dominated front, the solution with the higher crowding distance is preferred. On the other hand, if two solutions do not belong to the same non-dominated front, the one which belongs to a lower front is preferred (see [8]).

15.3.1.2 Adaptation to Our Problem

We use a binary-coded NSGA-II to solve the “upper level” subproblem of selecting the distribution centers to be opened. Each gene in the chromosome representing a solution is associated with a potential distribution center. Gene j , corresponding to a potential distribution center at location j , attains the value 1 if this distribution center is opened, and the value 0 otherwise. Thus, in terms of the decision variables of (15.1)–(15.18), the value of gene j is just $\sum_k z_{jk}$.

The genetic operators we use in this algorithm are selection, crossover and mutation. Since by the encoding above, the upper-level subproblem of our problem is a usual (bi-objective) binary vector optimization problem, we apply the mentioned genetic operators in a standard way. For implementation details, see Sect. 15.4.

To obtain complete solutions to the first-stage problem, however, also delivery tours have to be determined, i.e., after the choice of locations for the distribution centers, a capacitated VRP has to be solved. This yields then both the decision variables z_{jk} determining which vehicle k visits the distribution center in site j , and

the decision variables x_{ijk} describing the tours. Note that the third class of first-stage decision variables, the variables y_{ij} , result directly from the choice of the distribution center locations. The determination of the delivery tours will be described in the next subsection.

15.3.2 Savings Algorithm

As we use a heuristic approach to address the upper level decision on the distribution centers to be opened, we also apply a heuristic algorithm for tackling the routing problem on the lower decision level. We chose the well-known *Savings Algorithm* by Clarke and Wright. In terms of our application, this heuristic solution procedure for capacitated VRPs can be described as follows (cf. [11]). First, an initial solution is obtained by assigning each distribution center to be visited to a single tour: For the distribution center in village i , the associated route is $(0, i, 0)$, where 0 is the depot. By combining some of these forth-and-back tours through links (i, j) , savings regarding the tour length can be achieved. The savings value s_{ij} for link (i, j) with $i < j$ is calculated as $s_{ij} = d_{0i} + d_{0j} - d_{ij}$. Savings values are sorted in decreasing order. In each iteration, the best (i.e., largest) element s_{ij} for which the constraints below are satisfied is chosen and the two tours containing i and j are merged along link (i, j) to a single tour. The constraints are:

- Villages i and j are not on the same tour.
- In their respective tours, both village i and village j are adjacent to the depot.
- By merging the two tours, capacity constraints are not violated.

After performing this operation, link (i, j) is eliminated from the list and the iteration starts again until there are no savings left. In our case, we change this standard termination condition to the condition that exactly $|K|$ tours are achieved, corresponding to the $|K|$ available vehicles.

15.3.3 Sampling Procedures

15.3.3.1 General Remarks

After the first-stage decision has been made, i.e., the decision variables x , y and z have been fixed, the realization of the random demand is observed and the second-stage decision (described by u) is determined by solving the second-stage problem. As argued in [51], the solution of the second-stage problem (15.14)–(15.18) is straightforward: Each variable u_{jk} is chosen as large as possible given (15.15)–(15.17). Nevertheless, in our algorithmic context, there remains the question of how to evaluate a first-stage solution (x, y, z) produced by the NSGA-II and the Savings Algorithm with respect to objective function $f_2 = \mathbf{E}(R(y, z, \xi))$ (objective func-

tion f_1 is not problematic, since it is deterministic). We rely on *scenario-based* approaches, estimating the objective function value of f_2 by *sampling*.

Most works applying these approaches use a sample of scenarios fixed in advance. This is the so-called *fixed-sample* method, which we shall investigate as our first sampling variant, abbreviated by **Fixed**, to deal with the randomness in the demand. The solutions obtained from the NSGA-II are evaluated here based on the average objective function value over the sample drawn in advance. It is clear that in this way, the “true” probability distribution of the demand is replaced by an approximation.

In addition to that, we shall also study two other variants of dealing with the randomness, both working with a *variable* sample of scenarios, i.e., with a sample that is changed during the execution of the bi-objective optimization algorithm. One of them, our second sampling variant which we shall abbreviate by **Variable**, is quite straightforward: In each iteration of the NSGA-II, we use a new, independently generated random sample of demand vectors. Again, the evaluation of a solution is then based on the average objective function value over the (current) sample.

The third variant we study is again of variable-sample type, but contrary to the second variant, it uses a higher degree of interaction with the bi-objective optimization procedure and an archive of potentially efficient solutions. This is the Adaptive Pareto Sampling method, which we shall abbreviate by **APS**. In the single-objective stochastic optimization context, it has been shown that under certain conditions, variable-sample modifications of evolutionary algorithms converge to the true optimum (see, e.g., [27, 17]). In the multi-objective context, corresponding convergence results (convergence to the true Pareto front) are harder to obtain, but conditions can be given under which they hold for APS [19, 23].

15.3.3.2 Adaptive Pareto Sampling

The APS approach is an algorithmic framework for solving stochastic multi-objective optimization problems. It works by generating a series of random samples, solving the *deterministic* multi-objective optimization problems corresponding to them by any suitable subprocedure, and adapting the current approximation to the Pareto front iteratively based on the results from the calls of the subprocedure and on objective function evaluations with increasing precision.

To outline the algorithm for the bi-objective case, we assume a stochastic optimization problem of the form

$$\min (F_1(x), F_2(x)) \quad \text{s.t.} \quad x \in S \quad (15.19)$$

with

$$F_r(x) = \mathbf{E}(f_r(x, \omega)) \quad (r = 1, 2),$$

where S is the feasible set, f_r is the r -th cost function, ω represents the influence of randomness, and \mathbf{E} denotes the mathematical expectation. The expectation $\mathbf{E}(f_r(x, \omega))$ can be approximated by drawing N independent random scenar-

ios $\omega_1, \dots, \omega_N$. Then we can calculate the sample average estimate of $F_r(x) = \mathbf{E}(f_r(x, \omega))$ as

$$\frac{1}{N} \sum_{v=1}^N f_r(x, \omega_v) \approx \mathbf{E}(f_r(x, \omega)) \quad (r = 1, 2). \quad (15.20)$$

A deterministic problem corresponding to Eq. (15.19) can be solved to obtain an approximation of the solution. This deterministic approximation results by replacing the original objective functions with the sample average estimates (15.20). The problem

$$\min \left(\frac{1}{N} \sum_{v=1}^N f_1(x, \omega_v), \frac{1}{N} \sum_{v=1}^N f_2(x, \omega_v) \right) \quad \text{s.t.} \quad x \in S \quad (15.21)$$

is called the *bicriteria sample average approximation* (BSAA) problem.

The APS algorithm operates as described below (Algorithm 2). The algorithm works iteratively and the current solution set $L^{(k)}$ is updated at the end of each iteration. In each run, we first determine a proposal for the solutions by solving the deterministic BSAA problem. This can be done either by an exact or a metaheuristic method. Then these solutions and the elements of the solution set $L^{(k-1)}$ are combined and evaluated on a new sample of scenarios. Solutions which are dominated are removed from the set. The respective sample sizes used in steps (a) and (b) are denoted by s_k and \bar{s}_k . The algorithm terminates when a predefined maximum number of iterations is reached. We obtain the current solution set $L^{(k)}$ as the desired approximation to the set of Pareto-optimal solutions.

In our model, we use the NSGA-II algorithm to solve the BSAA problem in the solution proposal step in part (a). Furthermore, we can simplify the APS algorithm for our purposes in an obvious way, considering that the first objective function in our problem is deterministic.

Algorithm 2 Adaptive Pareto Sampling

Procedure APS

```

initialize the solution set  $L^{(0)}$  as the empty set
for iteration  $k = 1, 2, \dots$  {
  (a) "solution proposal":
    draw a sample of scenarios  $\{\omega_1, \dots, \omega_{s_k}\}$  of size  $s_k$ 
    for the drawn sample, determine the Pareto-optimal set  $S^{(k)}$ 
      of the BSAA problem with sample size  $s_k$ 
  (b) "solution evaluation":
    for each  $x \in L^{(k-1)} \cup S^{(k)}$  and each  $r = 1, 2$  {
      determine an estimate of  $F_r(x)$ , based on a new sample of  $\sigma_k$  scenarios
    }
    obtain  $L^{(k)}$  as the set of efficient solutions in  $L^{(k-1)} \cup S^{(k)}$  according
      to the cost function estimates just determined
}

```

15.4 Case Study

We have tested the three sampling-based NSGA-II variants described above using real-world data from the region of Thiès, Senegal. First, we describe the parameters used for the implementation. Then, computational results will be presented and compared.

15.4.1 Implementation Details

15.4.1.1 General Assumptions

The region of Thiès consists of 32 rural communities. Each rural community contains several villages. We treat each rural community as a separate instance. Therein, the most important village, after which the rural community is named, serves as a depot, although this may not always be the largest village in terms of number of inhabitants. Two vehicles whose capacities are not constraining are available. The distances between the villages correspond to the driving costs. The share of people who are able to go to their closest distribution center is determined by the following step function:

$$\psi(d) = \begin{cases} 1 & \text{if } d \leq 6, \\ 0.5 & \text{if } 6 < d \leq 15, \\ 0 & \text{if } d > 15, \end{cases}$$

where the distance unit is 1 km. The capacity of each distribution center is assumed to be three times the baseline demand of the village in which the distribution center is located. The baseline demand w_i equals the number of inhabitants of village i . The costs for opening a distribution center are the same for each potential location. Regarding the demand distribution, the random variable ξ_i is a combination of a random baseline term for the whole region and a village-specific random correction term [51]: $\xi_i = \xi_{bas} - \beta_2 + 2\beta_2 Z_i$ with $\xi_{bas} = \bar{\xi} - \beta_1 + 2\beta_1 Z$, where $\bar{\xi}$ is a constant parameter, Z and Z_i ($i = 1, \dots, n$) are independent uniformly distributed random numbers between 0 and 1, and $\beta_1, \beta_2 > 0$ are constant parameters. As in [51], the values $c_j = 5000$, $\bar{\xi} = 1$, $\beta_1 = 0.5$ and $\beta_2 = 0.5$ were chosen.

15.4.1.2 Parameters for the Sampling Approaches

For the sample size in the variant *Fixed*, we have tested the value 1500. The variant *Variable* uses samples with growing sample sizes $10 + (t - 1)$, where t is the index of the current NSGA-II iteration. For the variant *APS*, a total number K of APS iterations has been carried out (each containing several NSGA-II iterations); for K , we have tested the values 5, 10, 15, 20, 25 and 30. The sample size s_k in APS iteration k has been chosen as independent of k and fixed to the value 200, and the

sample size σ_k has been chosen as growing according to $\sigma_k = \bar{\alpha} + \bar{\beta}(k - 1)$. The results below refer to the special choice $\bar{\alpha} = 10,000$ and $\bar{\beta} = 5000/K$. Whenever possible, the idea of cumulative samples is used to reduce computation time. This sample scheme starts, e.g., with a sample size of $\bar{\alpha}$ and appends in each iteration a new sample of size $\bar{\beta}$, computing the average of the considered function over the union of the samples by a weighted mean so that the effort already invested during previous iterations is not wasted [27].

We aimed at a fair comparison of the investigated algorithms by providing each of them with (approximately) the same computation time for each of the considered test instances. Since all algorithmic variants use the NSGA-II algorithm, the number of NSGA-II iterations was the parameter by which we were able to control the overall computation time. Different test instances were given slightly different time budgets, but all of them in the range of a few minutes per single run. For each instance and each algorithmic version (one version of *Fixed*, one version of *Variable* and six versions of *APS*), 20 runs with different seeds were carried out.

15.4.1.3 Parameters for the NSGA-II Algorithm

For the binary-coded NSGA-II, we used a one-point crossover and a bitwise mutation. The crossover probability has been chosen as $p_c = 0.8$ and the mutation probability for each gene in a chromosome is $p_m = 1/\ell$, where ℓ is the string length. These values are chosen for all test instances. They are in the range of commonly used values in other NSGA-II applications. For the selection operator, we use the method of tournament selection with a tournament size of 2. For the population size parameter \mathcal{N} , different values in the range between 100 and 200 have been used in dependence of the instance, based on a pre-test for a single seed value. As described above, the maximum number of generations, i.e., the number of NSGA-II iterations, was determined with the aim of producing comparable computation times.

15.4.2 Performance Measures

In order to evaluate the quality of the proposed non-dominated solutions obtained by the execution of the different algorithmic variants, we need suitable performance measures. The definition of the quality of an approximation to the set of Pareto-optimal solutions is difficult because several aspects play a role. There is general agreement on the opinion that the assessment of a multi-objective optimization metaheuristic is a multi-objective problem itself. For the performance comparison of our different solution approaches, we use the following quantitative measures:

- the *hypervolume metric* defined in [53],
- a metric (originally proposed by Van Veldhuizen [52]) given by the ratio of points in the approximation set that are non-dominated points, and
- the *spacing metric* introduced by Schott [45].

The hypervolume metric Q_H requires defining a fixed reference point x_R in the objective space with the property that in each of the objective functions, x_R is not better than any feasible solution. The hypervolume metric Q_H is then defined as the hypervolume of the set of all those points that are dominated by the approximation set and that dominate the reference point. Larger hypervolumes are preferred.

The ratio metric Q_R suggested in [52] (denoted as Q_3 in Jaszkievicz [29]) is defined as

$$Q_R(A) = \frac{|A \cap T|}{|A|},$$

where A is the approximation set (the set of solutions proposed as non-dominated by the algorithm), and T is the true Pareto set. If one does not know the true Pareto set, a common procedure is to approximate T by a reference set \tilde{T} which consists of the non-dominated solutions within the union of all solution sets provided by any of the investigated algorithms.

The spacing metric Q_S for a bi-objective optimization problem, finally, is calculated as

$$Q_S = \left[\frac{1}{|A| - 1} \sum_{x \in A} (\bar{D} - D(x))^2 \right]^{1/2}$$

where A is again the approximation set,

$$D(x) = \min_{y \in A} \{|f_1(x) - f_1(y)| + |f_2(x) - f_2(y)|\},$$

and \bar{D} denotes the mean of all values $D(x)$ ($x \in A$). If Q_S is zero, the solutions in A are equally spaced from each other, which would be the ideal case.

As usual, we performed the calculations of the metrics based on normalized objective function values that lie in the interval $[0, 1]$. Ideally, the objective functions f_1 and f_2 should be transformed by

$$\tilde{f}_i(x) = (f_i(x) - \min_y f_i(y)) / (\max_y f_i(y) - \min_y f_i(y)).$$

However, the determination of $\max_y f_i(y)$ and $\min_y f_i(y)$ may be difficult, so in practice, surrogates for these two values are used. We determine an approximation to the worst case point $(\max_y f_1(y), \max_y f_2(y))$ by executing all solution algorithms, maximizing over the values found and adding a safety increment. The worst case point is also used as the reference point for the hypervolume metric. The ‘‘utopia point’’ $(\min_y f_1(y), \min_y f_2(y))$ is obtained in our case by setting both objective function values to zero.

15.4.3 Test Instances

The different algorithmic variants described above are tested on several problem instances. The test instances are taken from [51], where they were treated by a

branch-and-cut algorithm based on a fixed-sample approach with sample size 10. In this paper, we only consider the harder of these instances, namely those instances for the solution of which the necessary CPU time reported in [51] exceeded 3600 s, or which could not even be solved to optimality³ at all by the exact algorithm within the CPU time limit of 3 days chosen in [51]. In Table 15.1, the resulting 12 Senegal instances are listed, and their respective problem sizes (ranging from 18 to 31 nodes) are indicated.

Table 15.1 Senegal instances and problem instance sizes

No. Instance	# nodes
1 Diender Guedj	22
2 Fandane	24
3 Fissel	21
4 Mboro	31
5 Ndiagagniao	24
6 Ndiass	18
7 Ngandiouf	19
8 Notto	28
9 Pire Goureye	18
10 Pout	29
11 Tassette	21
12 Touba Toul	20

15.4.4 Computational Results

The programming language C has been used for the implementation of the algorithms. The programs have been executed on a computer with a 2.00 GHz Intel Core 2 Duo T7250 processor.

15.4.4.1 Performance Evaluation

For the performance assessments, the single solutions provided (as part of an approximation set) by each of the algorithms were post-evaluated with respect to objective function f_2 on a new sample of size $3 \cdot 10^4$; this sample was chosen independently from those used in the algorithms. This allows a fair and sufficiently accurate evaluation of Q_H and Q_S for the produced approximation sets, as well as

³ Note that “solved to optimality” in the sense of [51] does not mean that the true Pareto front of the original stochastic bi-objective problem has been found, but only that the true Pareto front of the bi-objective problem obtained by replacing the original probability distribution with the empirical distribution in the random sample has been determined. Only for this modified problem, the branch-and-cut algorithm is an “exact” solution algorithm.

an estimation of the ratio Q_R of non-dominated solutions among all solutions in the approximation set A .

The performance of each of the three solution approaches with respect to the hypervolume metric is shown in Table 15.2. In addition, for comparison purposes, the second column contains the hypervolumes achieved by the mathematical programming approach used in [51], based on sample size 10. This variant is denoted by MP Fixed 10. It provides the exact solution of the problem based on a sample of size 10 for those instances where the implementation in [51] was able to find it within 3 days, or, in the case of the other instances, the result of the MP-based heuristic proposed in [51]. Columns 3–5 contain the average hypervolumes achieved by our NSGA-II implementations for Fixed (with sample size 1500), Variable, and APS, respectively. As a representative for the six different tested parametrizations of APS, we took APS 10, i.e., the variant with $K = 10$ APS iterations. (Some observations on the influence of K on the performance of APS will be given later.) As stated above, we executed 20 runs for each of the NSGA-II-based algorithms in order to take account of the inherently stochastic nature of the NSGA-II. The reported hypervolume value is the average over these runs. For the deterministic algorithm MP Fixed 10, only one run was carried out.

The best values are printed in bold. We see that in 9 out of the 12 instances, APS 10 provided the best solutions. Nevertheless, the differences of the average hypervolume between the NSGA-II-based algorithms are very small, so we cannot say that a practically relevant superiority of APS with respect to the hypervolume is shown by the experimental outcome. This is underlined by statistical tests on the level of the single instances: For each instance, we tested the null hypothesis that the respective best NSGA-II-based algorithm has the same hypervolume as the second-best NSGA-II-based algorithm, against the alternative hypothesis that the hypervolumes differ. Welch's t-test (independent two-sample test, equal variances not assumed) was used. In none of these comparisons, the difference turned out as statistically significant at level $\alpha = 0.05$.

It is interesting to see that in spite of much larger computation times (see Table 15.5), the mathematical programming solution approach of Tricoire et al. [51] produces worse results than each of the three considered alternatives in 11 out of the 12 instances.⁴ Of course, this is due to the only small sample size of 10 it can build upon in view of runtime limitations, whereas the NSGA-II variants used in this paper can apply much higher sample sizes. We can interpret this result by saying at the given problem, it obviously pays off to invest runtime rather in sampling (even at the price of solving the deterministic counterpart problem only heuristically) than in solving the deterministic counterpart problem exactly.

⁴ Thus, by a binomial test over the 12 instances, it performed significantly worse in total at a significance level of $\alpha = 0.05$.

Table 15.2 Hypervolumes Q_H for MP Fixed 10, Fixed 1500, Variable, and APS 10 (averaged over 20 runs for the GA-based algorithms). Best values are in bold

Test instance no.	MP Fixed 10	Fixed 1500	Variable	APS 10
1	0.834287	0.835609	0.835680	0.835680
2	0.704308	0.705690	0.705689	0.705689
3	0.788490	0.800320	0.800332	0.800333
4	0.731014	0.736246	0.736523	0.736533
5	0.827665	0.836296	0.836295	0.836299
6	0.772287	0.772505	0.772586	0.772589
7	0.732510	0.732214	0.732213	0.732214
8	0.768785	0.790878	0.790947	0.790953
9	0.817639	0.817783	0.817787	0.817787
10	0.850377	0.856147	0.856163	0.856167
11	0.804675	0.804685	0.804789	0.804794
12	0.778684	0.778942	0.778942	0.778939

Table 15.3 shows the results for the measure Q_R , the ratio of proposed solutions that turn out as actually efficient in the post-evaluation. We see that in this evaluation measure, the APS variant is clearly superior compared to all other algorithms. Firstly, it produces the best average Q_R scores in all 12 instances. Secondly, the superiority is statistically significant (usually even highly significant) for 75% of all instances. It is not surprising that APS 10 outperforms the other algorithms with respect to the criterion that proposed solutions should be Pareto-optimal indeed, since by construction, the algorithm eliminates dominated solutions from the archive in each iteration. What comes less expected is that this desirable property does not impair the ability of APS 10 to achieve a competitive hypervolume within the given computation time as well, as it has been seen in Table 15.2.

Table 15.3 Ratios Q_R of truly efficient solutions for MP Fixed 10, Fixed 1500, Variable, and APS 10 (averaged over 20 runs for the GA-based algorithms). Best values are in bold. One, two and three stars denote a significant difference to the second-best algorithm at the considered instance over the 20 runs for significance level 0.05, 0.01 and 0.001, respectively

Test instance no.	MP Fixed 10	Fixed 1500	Variable	APS 10
1	0.70588	0.95079	0.94451	0.99441 ***
2	0.88235	0.95633	0.94537	0.98444 **
3	0.64486	0.96123	0.95721	0.99535 ***
4	0.74107	0.99400	0.98900	1.00000 *
5	0.66304	0.94297	0.96479	0.98676 ***
6	0.85455	0.97487	0.96255	0.99825 ***
7	0.77333	0.97634	0.97934	0.98954
8	0.83720	0.96455	0.96183	0.99149 ***
9	0.92727	0.99007	0.98341	0.99917 **
10	0.83636	0.95715	0.96382	0.99468 ***
11	0.87879	0.98190	0.98305	0.99290
12	0.90698	0.99694	0.99485	1.00000

Table 15.4 shows the resulting values of the spacing metric. With respect to this evaluation measure, no clear performance hierarchy between the methods can be observed. It should be noted that as long as the values of the spacing metric Q_S do not differ too much for different algorithms under consideration, the spacing metric is not the most important criterion for ranking the algorithms, since the true Pareto front can obviously have a value of Q_S that is distinctly larger than zero, whereas a poor approximation to it can have a value of Q_S closer to zero. We see in Table 15.4 that in the case of the majority of instances, the Q_S values are very similar for all algorithms, which means that the degree to which the solution points are evenly distributed is visually almost indistinguishable between the algorithms.

From Table 15.5 we see that all heuristic methods are able to solve each of the test instances within a few minutes, which even holds for instances that cannot be solved by the mathematical programming algorithm within 3 days.

Table 15.4 Spacing metric Q_S for the algorithms MP Fixed 10, Fixed 1500, Variable, and APS 10 (averaged over 20 runs for the GA-based algorithms). Best values are in bold

Test instance no.	MP Fixed 10	Fixed 1500	Variable	APS 10
1	0.02756	0.02811	0.02667	0.02673
2	0.01301	0.01375	0.01436	0.01353
3	0.00766	0.00858	0.00824	0.00838
4	0.02143	0.01722	0.01702	0.01718
5	0.01196	0.01139	0.01127	0.01122
6	0.02903	0.02595	0.02720	0.02687
7	0.01442	0.01356	0.01356	0.01357
8	0.02078	0.01493	0.01681	0.01634
9	0.00872	0.00691	0.00693	0.00669
10	0.01233	0.00973	0.00990	0.00935
11	0.01933	0.02147	0.02096	0.02109
12	0.02737	0.02555	0.02556	0.02550

Table 15.5 Computation times per single run in seconds. MPFx10 = MP Fixed 10, Fx1500 = Fixed 1500, Var = Variable

No.	MPFx10	Fx1500	Var	APS5	APS10	APS15	APS20	APS25	APS30
1	> 3 days	122	119	123	125	126	126	128	124
2	> 3 days	418	404	401	403	411	409	409	404
3	> 3 days	261	259	257	255	252	250	252	267
4	> 3 days	546	523	520	512	526	528	519	518
5	> 3 days	252	251	258	261	264	268	267	259
6	6925	90	89	91	93	92	92	93	96
7	13262	286	288	289	292	290	291	287	282
8	> 3 days	352	349	346	349	348	354	354	336
9	3663	85	83	80	80	85	85	82	88
10	> 3 days	575	569	564	557	549	548	565	554
11	4974	123	119	122	123	123	123	124	125
12	12707	105	104	103	102	103	103	103	103

15.4.4.2 Behavior of APS for Increasing Number of Iterations

In the previous subsection, we compared the behavior of APS to the alternative algorithms for a fixed choice of $K = 10$ APS iterations. A more detailed analysis should address the question how the number K influences the quality of the algorithm according to our chosen evaluation measures. We focus the analysis on the measures Q_H and Q_R . It can be expected that for large K , the quality will deteriorate since the NSGA-II, executed as a subprocedure, will not receive anymore a sufficiently large number of iterations to work efficiently. Recall that the overall runtime budget is always kept (approximately) fixed in order to enable a fair comparison.

In Figs. 15.1, 15.2 and 15.3 the three special instances Fissel, Ndiagniao and Tassette are used to illustrate the development of the hypervolume and of the ratio of truly efficient solutions when the number of APS iterations is successively increased and the number of NSGA-II iterations in the subprocedure is correspondingly decreased. We see that as far as the hypervolume is concerned, an optimal value is reached already by comparably few APS iterations. For more than about 20 iterations, the performance starts distinctly to drop (first plots in Fig. 15.1). A converse effect can be observed for the share Q_R of the efficient solutions among all provided solutions (second plots in Fig. 15.1): this share seems to be continuously improving with growing APS iteration number.

We verified the last observation by carrying out a linear regression analysis for each instance (independent variable: number K of APS iterations, dependent variable: Q_R) and by testing whether the obtained regression coefficients are significantly positive. The results are shown in Table 15.6. We see that in 9 out of the 12 instances, the regression coefficient is larger than zero, in 2 instances it is smaller than zero. According to a binomial test, this verifies that the probability of a positive regression coefficient is significantly higher than that of a negative regression coefficient ($\alpha = 0.05$). Furthermore, by a statistical F-test, we tested the null hypothesis that the regression coefficient is zero against the alternative hypothesis that it is different from zero for each instance separately. In the case of four instances (all with positive regression coefficient), the null hypothesis was rejected. In the two instances with negative regression coefficient, the null hypothesis was not rejected, i.e., the negativity of the regression coefficient is not significant. Together, this provides a good statistical justification for the conjecture that with an increasing iteration number of APS (in the range between 5 and 30), the share of efficient solutions among all provided solutions will increase.

From the results, it is obvious that an APS iteration in the range of 10–20 achieves a good compromise between the aim of maximizing Q_H and the aim of maximizing Q_R . In this range, the value of Q_H has typically not yet relevantly dropped, and on the other hand, the value of Q_R is typically already higher than for very small K values.

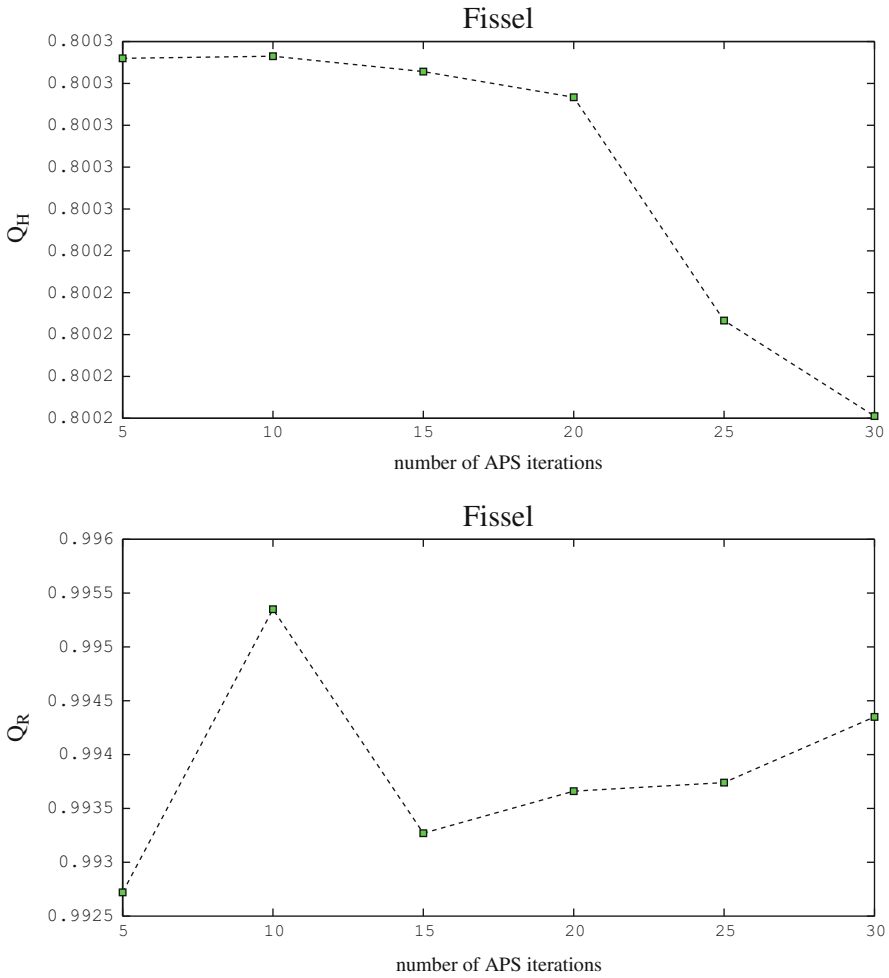


Fig. 15.1 Development of the hypervolume Q_H and the ratio of efficient solutions Q_R for growing iteration numbers of APS (overall runtime fixed) in the case of Fissel instances. First plots show Q_H , second plots show Q_R

15.4.4.3 Graphical Comparison

In order to get a better insight into the behavior of the algorithms studied in this paper and to show how they perform in approximating the true set of Pareto-optimal solutions, a look at Fig. 15.4 can be instructive. The figure visualizes the non-dominated fronts achieved by the different solution approaches for the instance Notto. For this instance, the branch-and-cut-based algorithm from [51] considering 10 scenarios is not able to find all non-dominated solutions within the time limit of 3 days. Under MP Fixed 10, we depict the solutions provided by this algorithm within

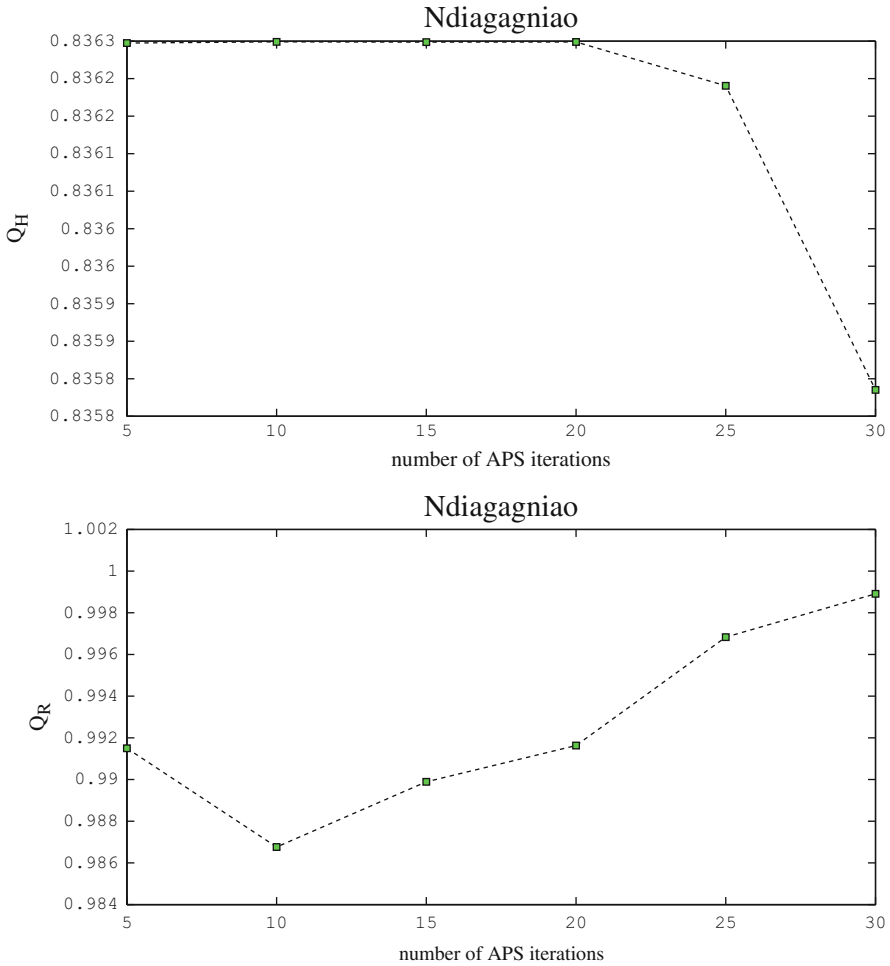


Fig. 15.2 Development of the hypervolume Q_H and the ratio of efficient solutions Q_R for growing iteration numbers of APS (overall runtime fixed) in the case of Ndiagagniao instances. First plots show Q_H , second plots show Q_R

the given time (these solutions are Pareto-optimal with respect to the chosen sample, but the solution set is incomplete even with respect to the sample). Within very small runtime, the NSGA-II-based solution procedures are able to find additional solutions close to the right end of the Pareto front (high costs, but low uncovered demand) and extend in this way the options of the decision maker to solutions that effect nearly full coverage or full coverage of the demand. Also in that range of the front for which MP Fixed 10 yielded points, the applied heuristic techniques deliver high-quality solutions. Already the NSGA-II-based variant Fixed 10 using only a sample size of 10 is surprisingly good.

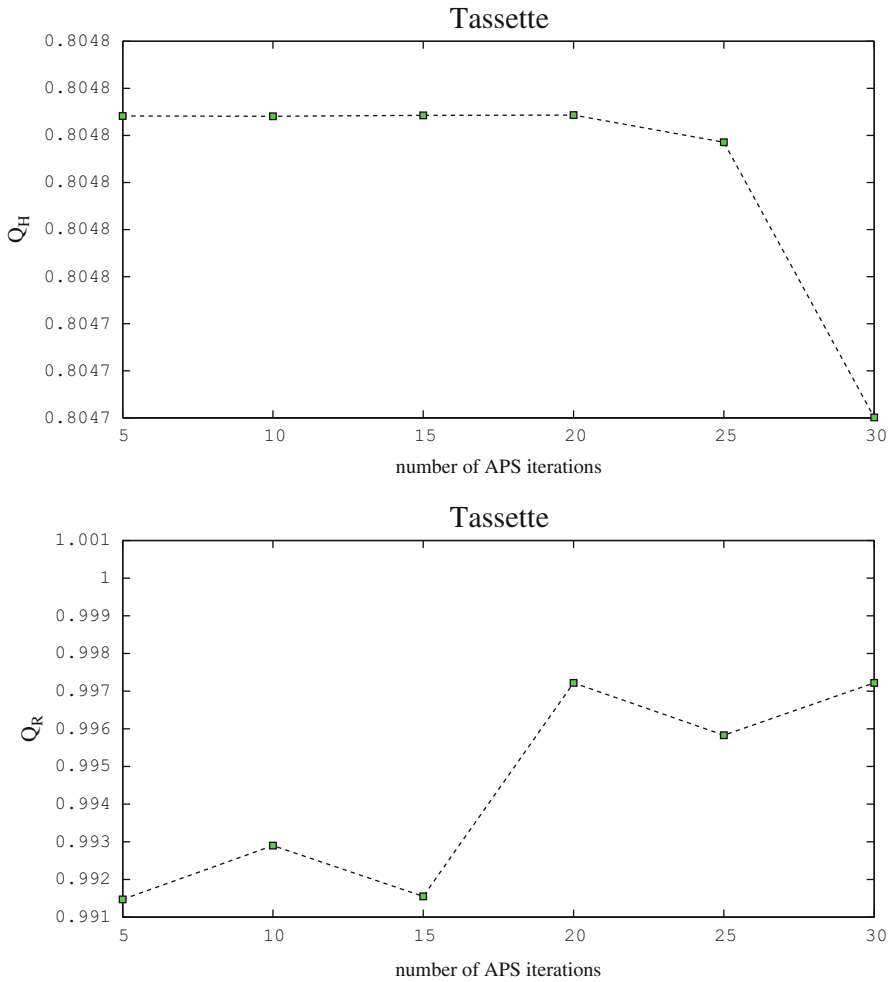


Fig. 15.3 Development of the hypervolume Q_H and the ratio of efficient solutions Q_R for growing iteration numbers of APS (overall runtime fixed) in the case of Tassette instances. First plots show Q_H , second plots show Q_R

15.5 Conclusions

In this paper, we presented different approaches for solving an extended version of the covering tour problem (CTP), namely the bi-objective stochastic CTP proposed in [51]. In [51], problem instances from Senegal have been solved by applying branch-and-cut within an epsilon-constraint algorithm to a (small) fixed random sample of scenarios. Since larger problem instances cannot be solved within reasonable time by this approach, we used in the present paper the metaheuristic

Table 15.6 Regression analysis for the dependence of Q_R on the number K of APS iterations. A star marks instances for which the regression coefficient is significantly different from zero at significance level $\alpha = 0.05$

Test instance no.	Regression coefficient
1	0.0003899
2	0.0003537*
3	0.0000212
4	0.0000549
5	0.0003942*
6	-0.0000453
7	0.0000020
8	0.0003327*
9	0.0000325
10	-0.0000059
11	0.0002469*
12	0.0000000

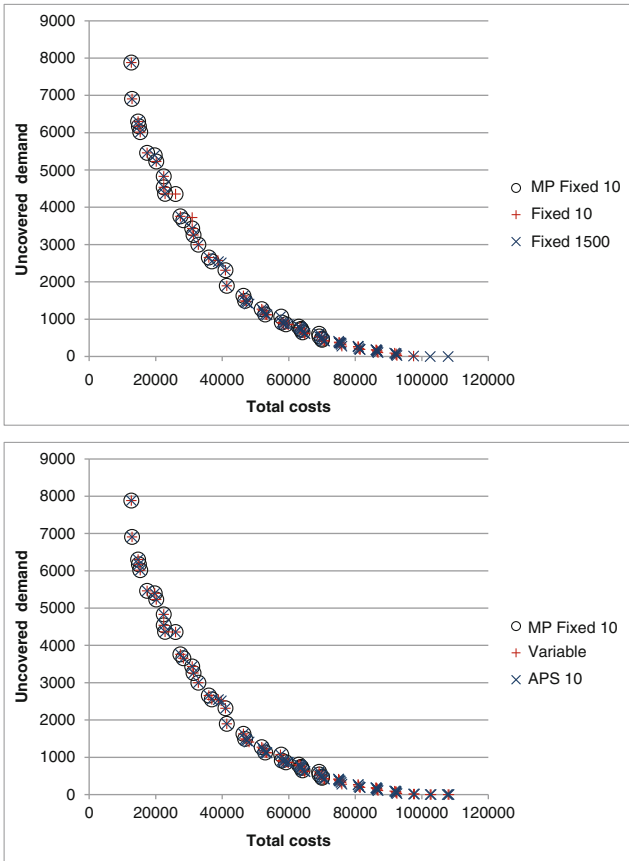


Fig. 15.4 Pareto front approximations provided by different solution approaches for instance Notto

NSGA-II algorithm to tackle the multi-objective aspect of the problem. As a way to cope with the stochastic aspect, we applied different sampling methods: (i) a variant where a fixed random sample is used, (ii) a variant where the sample is exchanged in each NSGA-II iteration, and (iii) the more sophisticated Adaptive Pareto Sampling (APS) technique which works with an archive of solutions and iteratively proposes and tests new candidate solutions for inclusion in the current approximation to the Pareto set.

Our experimental results, carried out for the same test benchmark as it has been studied in [51], led to two essential insights: Firstly, at the considered problem, it pays off to invest computation time rather in a more extensive sampling than in an *exact* solution of the deterministic sample average optimization problem obtained by applying the optimization procedure to a chosen random sample. The quality of our NSGA-II-based solution sets was consistently better than that of the solution sets found in [51], though we needed only a small fraction of the computation time invested there. Secondly, all three investigated NSGA-II-based variants performed very well with respect to the hypervolume measure (with only a slight and statistically not significant advantage observable for APS with small iteration number), but they differed by a second evaluation measure, the share Q_R of efficient solutions among the solutions *proposed* as efficient: in this respect, APS performs better than the two other variants. In terms of the number K of APS iterations, we observed a trend of increasing Q_R with increasing K , expressed by a usually positive regression coefficient. On the other hand, the number K of APS iterations should not be chosen too large since beyond a value of about $K = 20$, the achieved hypervolume starts to drop. This suggests the use of APS with a moderate number of iterations for achieving a good compromise between the two aims of reducing the area between approximated and true Pareto front, and of producing only solutions that have a good chance of being Pareto-optimal (or nearly Pareto-optimal) indeed. All comparisons have been done based on (almost) equal computation times, which means that as the number of APS iterations has been increased, the number of NSGA-II iterations used in the subprocedure has been decreased correspondingly.

For the larger instances from the Senegal test benchmark for which the Pareto front could not be determined completely in [51] within 3 days, all of the proposed heuristic algorithms were able to find, within a runtime that is pre-determined by only a few minutes, a broader Pareto front approximation containing also additional non-dominated solutions on that side of the front that has turned out as hardly accessible by the mathematical programming approach. In total, all applied heuristic solution implementations produced high-quality solutions and should be considered as well-suited for solving the optimization problem under investigation.

Our three NSGA-II-based algorithms are “anytime” algorithms, i.e., their execution can be arbitrarily prolonged, and the solution quality can be expected to increase with growing runtime. An advantage of APS over the more straightforward variable-sample variant is that by construction, APS aims at the possibility of convergence to the true Pareto front, and on certain conditions on the problem, this convergence can be proven to hold in a rigorous mathematical sense (see [19, 21, 23]). Up to

now, however, we did not yet manage to verify this mathematical property for the problem under consideration in this paper.

Several issues for future research remain open. In the following remarks, we focus on the technical aspect and do not address possible further research concerning the envisaged application in disaster management, which would of course also be an interesting and important direction. Concerning the genetic algorithm component, an open question is whether and how the genetic operators can be modified to improve the performance of the NSGA-II for the purposes of inclusion within a sampling framework. To give an example, it might be promising to implement an adaptive mutation operator as proposed by Carvalho and Araujo [6] where the strength of the mutation depends on information about the population. In the sampling framework, not only the diversity of the population, but also statistical characteristics of the current accuracy of the objective function estimation might be taken into account. A second issue is the use of a so-called non-dominated tree, as proposed in Mendes and de Vasconcelos [38]. Furthermore, probabilistic dominance relations may be used, as it has been proposed in [12, 28, 48].

With respect to the variable-sample approach and APS, a solution-dependent adaptation of the sample size after each iteration (cf. [27]) could be a promising topic for further experiments. This might lead to algorithmic approaches similar to racing algorithms (cf. [2]). Moreover, whereas the investigations in this paper concern the bi-objective situation, results on related problems with three or more objectives would be very valuable.

From a more theoretical point of view, finally, it would be interesting to obtain *analytical* results on the questions addressed in this paper. For example, one would like to know (at least for simplified stochastic multi-objective problems) under which circumstances variable-sample techniques outperform fixed-sample techniques and vice versa in an asymptotic consideration.

Acknowledgements We want to express our thanks to Fabien Tricoire for his help during the preparation of this paper.

References

1. L. Amodeo, C. Prins, D. Sanchez, Comparison of metaheuristic approaches for multi-objective simulation-based optimization in supply chain inventory management, in *Applications of Evolutionary Computing*. Lecture Notes in Computer Science (Springer, Berlin, 2009), pp. 798–807
2. P. Balaprakash, M. Birattari, T. Stützle, M. Dorigo, Estimation-based metaheuristics for the probabilistic traveling salesman problem. *Comput. Oper. Res.* **37**, 1939–1951 (2010)
3. R. Baldacci, M.A. Boschetti, V. Maniezzo, M. Zamboni, Scatter search methods for the covering tour problem, in *Metaheuristic Optimization via Memory and Evolution*, ed. by R. Sharda et al. (Springer, Berlin, 2005), pp. 59–91
4. M. Basseur, E. Zitzler, Handling uncertainty in indicator-based multiobjective optimization. *Int. J. Comput. Intell. Res.* **2**, 255–272 (2006)

5. F. Ben Abdelaziz, Solution approaches for the multiobjective stochastic programming. *Eur. J. Oper. Res.* (2011). doi:10.1016/j.ejor.2011.03.033
6. A.G. Carvalho, A.F.R. Araujo, Improving NSGA-II with an adaptive mutation operator, in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference, July 2009, Montreal* (2009), pp. 2697–2700
7. J.R. Current, D.A. Schilling, The median tour and maximal covering tour problems: formulations and heuristics. *Eur. J. Oper. Res.* **73**(1), 114–126 (1994)
8. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
9. H. Ding, L. Benyoucef, X. Xie, A simulation-based multi-objective genetic algorithm approach for networked enterprises optimization. *Eng. Appl. Artif. Intell.* **19**, 609–623 (2006)
10. K. Doerner, A. Focke, W.J. Gutjahr, Multicriteria tour planning for mobile healthcare facilities in a developing country. *Eur. J. Oper. Res.* **179**(3), 1078–1096 (2007)
11. W. Domschke, *Logistik: Rundreisen und Touren*, 3rd edn. (Oldenbourg, München, 1990)
12. H. Eskandari, C.D. Geiger, Evolutionary multiobjective optimization in noisy problem environments. *J. Heuristics* **15**, 559–595 (2009)
13. H. Eskandari, L. Rabelo, M. Mollaghasemi, Multiobjective simulation optimization using an enhanced genetic algorithm, in *Proceedings of the 37th Conference on Winter Simulation, WSC '05*, Orlando, FL (2005), pp. 833–841
14. F. Foord, Gambia: evaluation of the mobile health care service in West Kiang district. *World Health Stat. Q.* **48**(1), 18–22 (1995)
15. M. Gendreau, G. Laporte, F. Semet, The covering tour problem. *Oper. Res.* **45**(4), 568–576 (1997)
16. L. Gustavsson, Humanitarian logistics: context and challenges. *Forced Migr. Rev.* **18**, 6–8 (2003)
17. W.J. Gutjahr, A converging ACO algorithm for stochastic combinatorial optimization, in *Proceedings of SAGA 2003 (Stochastic Algorithms: Foundations and Applications)*. Lecture Notes in Computer Science, vol. 2827 (Springer, Berlin, 2003), pp. 10–25
18. W.J. Gutjahr, Two metaheuristics for multiobjective stochastic combinatorial optimization, in *Proceedings of SAGA 2005 (Stochastic Algorithms: Foundations and Applications)* (Springer, Berlin, 2005), pp. 116–125
19. W.J. Gutjahr, A provably convergent heuristic for stochastic bicriteria integer programming. *J. Heuristics* **15**(3), 227–258 (2009)
20. W.J. Gutjahr, Recent trends in metaheuristics for stochastic combinatorial optimization. *Cent. Eur. J. Comput. Sci.* **1**, 58–66 (2011)
21. W.J. Gutjahr, Runtime analysis of an evolutionary algorithm for stochastic multi-objective combinatorial optimization. *Evol. Comput.* **20**, 395–421 (2012)
22. W.J. Gutjahr, A. Pichler, Stochastic multi-objective optimization: a survey on non-scalarizing methods. *Ann. Oper. Res.* (2013). doi:10.1007/s10478-013-1369-5
23. W.J. Gutjahr, P. Reiter, Bi-objective project portfolio selection and staff assignment under uncertainty. *Optimization* **59**, 417–445 (2010)
24. M.H. Ha, N. Bostel, A. Langevin, L.-M. Rousseau, An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. *Eur. J. Oper. Res.* **226**(2), 211–220 (2013)
25. M. Hachicha, M.J. Hodgson, G. Laporte, F. Semet, Heuristics for the multi-vehicle covering tour problem. *Comput. Oper. Res.* **27**(1), 29–42 (2000)
26. M.J. Hodgson, G. Laporte, F. Semet, A covering tour model for planning mobile health care facilities in Suhum district, Ghana. *J. Reg. Sci.* **38**(4), 621–638 (1998)
27. T. Homem-de-Mello, Variable-sample methods for stochastic optimization. *ACM Trans. Model. Comput. Simul.* **13**(2), 108–133 (2003)
28. E.J. Hughes, Evolutionary multi-objective ranking with uncertainty and noise, in *Proceedings of EMO '01 (Evolutionary Multicriterion Optimization)*, (Springer, Berlin, 2001), pp. 329–343

29. A. Jaskiewicz, Evaluation of multiple objective metaheuristics, in *Metaheuristics for Multi-objective Optimization*, ed. by X. Gandibleux et al. Lecture Notes in Economics and Mathematical Systems, vol. 535 (Springer, Berlin, 2004), pp. 65–89
30. M.T. Jensen, Reducing the run-time complexity of multiobjective EAs: the NSGA-II and other algorithms. *IEEE Trans. Evol. Comput.* **7**(5), 503–515 (2003)
31. N. Jozefowicz, A column generation approach for the multi-vehicle covering tour problem, in *Proceedings of the 12th ROADEF Conference*, Saint-Etienne, March 2011
32. N. Jozefowicz, A branch-and-price algorithm for the multiple vehicle covering tour problem, in *Proceedings of ODYSSEUS 2012*, Mykonos, May 2012
33. N. Jozefowicz, F. Semet, E.-G. Talbi, The bi-objective covering tour problem. *Comput. Oper. Res.* **34**(7), 1929–1942 (2007)
34. A. Konak, D.W. Coit, A.E. Smith, Multi-objective optimization using genetic algorithms: a tutorial. *Reliab. Eng. Syst. Saf.* **91**(9), 992–1007 (2006)
35. A. Liefvooghe, M. Basseur, L. Jourdan, E.-G. Talbi, Combinatorial optimization of stochastic multi-objective problems: an application to the flow-shop scheduling problem, in *Evolutionary Multi-Criterion Optimization*. Lecture Notes in Computer Science, vol. 4403 (Springer, Berlin, 2007), pp. 457–471
36. A. Liefvooghe, M. Basseur, L. Jourdan, E.-G. Talbi, ParadisEO-MOEO: a framework for evolutionary multi-objective optimization, in *Evolutionary Multi-Criterion Optimization*. Lecture Notes in Computer Science, vol. 4403 (Springer, Berlin, 2007), pp. 386–400
37. W.-K. Mak, D.P. Morton, R.K. Wood, Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Oper. Res. Lett.* **24**(1–2), 47–56 (1999)
38. J.B. Mendes, J.A. de Vasconcelos, Using an adaptation of a binary search tree to improve the NSGA-II nondominated sorting procedure, in *Proceedings of the 8th International Conference on Simulated Evolution and Learning*, Kanpur, December 2010, pp. 558–562
39. L. Motta, L.S. Ochi, C. Martinhon, GRASP metaheuristics to the generalized covering tour problem, in *Proceedings of the 4th Metaheuristics International Conference*, Porto, July 2001, pp. 387–391
40. Z. Naji-Azimi, J. Renaud, A. Ruiz, M. Salari, A covering tour approach to the location of satellite distribution centers to supply humanitarian aid. *Eur. J. Oper. Res.* **222**(3), 596–605 (2012)
41. P.C. Nolz, K.F. Doerner, W.J. Gutjahr, R.F. Hartl, A bi-objective metaheuristic for disaster relief operation planning, in *Advances in Multi-Objective Nature Inspired Computing*, ed. by C.A. Coello Coello, C. Dhaenes, L. Jourdan. Studies in Computational Intelligence, vol. 272 (Springer, Berlin, 2010), pp. 167–187
42. P.C. Nolz, F. Semet, K.F. Doerner, Risk approaches for delivering disaster relief supplies. *OR Spectr.* **33**(3), 543–569 (2011)
43. S. Rath, W.J. Gutjahr, A math-heuristic for the warehouse location routing problem in disaster relief. *Comput. Oper. Res.* **42**, 25–39 (2014)
44. S. Rath, M. Gendreau, W.J. Gutjahr, Bi-objective stochastic programming models for determining depot locations in disaster relief operations planning. *Int. Trans. Oper. Res.* **23**, 997–1023 (2016)
45. J.R. Schott, Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge (1995)
46. W. Swaddiwudhipong, C. Chaovakiratipong, P. Nguntra, P. Lerdlukanavongse, S. Koonchote, Effect of a mobile unit on changes in knowledge and use of cervical cancer screening among rural Thai women. *Int. J. Epidemiol.* **24**(3), 493–498 (1995)
47. A. Syberfeldt, A. Ng, R.I. John, P. Moore, Multi-objective evolutionary simulation-optimisation of a real-world manufacturing problem. *Robot. Comput. Integr. Manuf.* **25**, 926–931 (2009)
48. J. Teich, Pareto-front exploration with uncertain objectives, in *Proceedings of EMO '01 (Evolutionary Multicriterion Optimization)* (Springer, Berlin, 2001), pp. 314–328

49. A.S. Thomas, L.R. Kopczak, From logistics to supply chain management: the path forward in the humanitarian sector. Fritz Institute [online] (2005), available at: www.fritzinstitute.org/PDFs/WhitePaper/FromLogisticsto.pdf. Accessed 08 Oct 2012
50. A. Thomas, M. Mizushima, Logistics training: necessity or luxury? *Forced Migr. Rev.* **22**, 60–61 (2005)
51. F. Tricoire, A. Graf, W.J. Gutjahr, The bi-objective stochastic covering tour problem. *Comput. Oper. Res.* **39**(7), 1582–1592 (2012)
52. D.A. Van Veldhuizen, Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Ph.D. Thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH (1999)
53. E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithms—a comparative case study, in *Parallel Problem Solving from Nature V*, ed. by A.E. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel (Springer, Berlin, 1998), pp. 292–301

Chapter 16

A Metaheuristic Framework for Dynamic Network Flow Problems

M. Hajjem, H. Bouziri, and El-Ghazali Talbi

Abstract Dynamic network problems is a very interesting topic in modeling real life situations where we aim to send some flow to a given destination within time dependent parameters. This can occur in many applications such in evacuation of people or vehicles in emergency time.

The majority of existing algorithms are based on mathematical approximations. However, this work proposes another technique based on metaheuristics. A general framework is provided in both single and population based algorithms. Therefore, basic search techniques are proposed such as the crossover or the mutation. Moreover, solution representations are given within a general metaheuristic scheme. In addition, we assess a genetic algorithm by an experimental study is conducted on a case study of a building evacuation.

Keywords Metaheuristic • Genetic algorithm • Dynamic flow problem • Evacuation

16.1 Introduction

Network flow is an important topic in combinatorial optimization arising in various applications such as evacuation, transportation, telecommunication and finance. We distinguish between static and dynamic approaches. Static network flow problems have been known for many years as valuable tools to model various real life applications. Deep theorems and efficient polynomial algorithms have been developed. However, these models fail to capture the dynamic property of several real problems such as evacuation problem, road or air traffic control, production systems and

M. Hajjem
LARODEC, ISG, University of Tunis, Tunis, Tunisia

H. Bouziri (✉)
LARODEC, ESSEC, University of Tunis, Tunis, Tunisia
e-mail: hend.bouziri@gmail.com

E.G. Talbi
INRIA Laboratory, CRISTAL/CNRS, University Lille 1, Villeneuve d'Ascq, France

communication networks. Time in these problems is an essential component, either because the flow takes time to pass from one location to another, or because the structure of the network changes over time.

Ford and Fulkerson [14] have proposed dynamic network flow models (named also flow over time) which are more appropriate than static one to simulate several real life applications. Considering flow over time, flow does not travel instantaneously from one node to another but it requires a certain amount of time to cross an arc. Hence, in dynamic network, each arc is characterized by a transit time which computes the time needed to travel this arc. flow over time allows the introduction of new problems and techniques [26]. When transit time of arcs are either flow-dependent or time-dependent, flow over time problems become more difficult [11]. Our idea is to use metaheuristics that have the reputation to be efficient in solving such difficult problems.

The main contribution of this chapter is to provide a metaheuristic framework to solve flow over time problems. The next section recalls the basic concepts of static network flow models. The third section describes the dynamic network, its characteristics and its variants. The fourth section focus on the definition of dynamic flow models and their complexities. Section 16.5 gives an overview of metaheuristic techniques. Section 16.6 explains the design of metaheuristic for dynamic network flow problems. Finally in Sect. 16.7, we present a case study where we adapt the proposed framework for designing an evolutionary schema to deal with evacuation problem . Section 16.8 is devoted to main conclusions and research perspectives.

16.2 Basic Notions and Results of Static Network Flow Problems

Static network is a directed graph $G = (N, A)$ where N is the set of nodes and A is the set of arcs. Each arc $a \in A$ is characterized by the capacity c_a . Each static network G defines three type of nodes: source nodes, sink nodes and intermediate nodes.

Given a static network G , the static flow function $x : A \rightarrow \mathbb{R}^+$ assigns to each arc a a non negative flow denoted by x_a . This flow must satisfy the following constraints:

$$x_a \leq c_a, \forall a \in A \quad (16.1)$$

$$\sum_{a \in \zeta^-(n)} x_a = \sum_{a \in \zeta^+(n)} x_a, \forall n \in N \setminus \{s, t\} \quad (16.2)$$

Equation (16.1) corresponds to *the capacity constraint* where the flow x_a should not exceed the fixed upper bound c_a . We say that x_a is *feasible* if it obeys this constraint. Whereas, Eq. (16.2) introduces *the flow conservation constraint* where $\zeta^-(n)$ and $\zeta^+(n)$ denote respectively the set of outgoing arcs of node n and incoming arcs into node n . This constraint implies that the total flow entering each node must be equal to the total flow leaving this node.

The value of flow defined by $val(X)$ presents the amount of flow that can pass from the source to the sink of a given network. This value of flow have to verify constraints (16.1) and (16.2). It is computed as follows:

$$\begin{aligned} \text{val}(X) &= \sum_{a \in \zeta^+(s)} x_a - \sum_{a \in \zeta^-(s)} x_a \\ &= \sum_{a \in \zeta^+(t)} x_a - \sum_{a \in \zeta^-(t)} x_a \end{aligned}$$

Two basic network flow problems are defined in literature. The first one is *the maximum flow problem* which attempts to achieve the maximum value of flow $\text{val}(X)$ from the source to the sink. Polynomial algorithms are provided to find the maximum flow in a given network. The most popular one was proposed by Ford and Fulkerson [13]. The second problem is *the minimum cost flow problem* which attempts to achieve the maximum flow with minimum cost. Similarly to the maximum flow problem, the minimum cost flow problem is polynomially solved [6].

These two basic problems have been treated and solved in different ways according to the type of the considered network. We distinguish four network flow problems:

- *The $s - t$ -flow problem* is defined on networks with a single source s and a single sink t .
- *The multiple source multiple sink problem* is introduced on networks with multiple sources and multiple sinks. This problem can be transformed into $s - t$ -flow problems by adding two artificial nodes: the super source which is connected to the set of sources and the super sink which is connected to the set of sinks.
- *The transshipment flow problem* is a variant of multiple sources and multiple sinks problem where supplies at sources and demands of sinks are fixed. This problem can be transformed to $s - t$ -flow problem with fixed supply on the super source and fixed demand on the super sink.
- *The multi-commodity problem* presents several types of flow in the same network. The object is to construct flows for the commodities that satisfy the demand for each commodity at each node without violating the constraints imposed by Eqs. (16.1) and (16.2).

16.3 Dynamic Network

Ford and Fulkerson [14] introduced the dynamic network by adding the time dimension to the classical network. In this way, a *transit time* is defined for every arc. Therefore, a dynamic network $G_T = (N, A, T)$ is a directed graph in which N is the set of nodes, A is the set of directed arcs and T is the time horizon.

16.3.1 Time Horizon

We distinguish between finite time horizon and infinite time horizon. In finite time horizon models, flow units should arrive to the destination node before a given time

T [31]. This means that only the amount of flow which arrives to the destination node before this fixed horizon time T is considered.

Infinite horizon time model does not have a specified and fixed period to solve problems [28]. It is used for example where activities are happen periodically like inventory problems.

The time horizon can be classified into discrete or continuous time. In discrete horizon time models [19], the horizon time T is broken up into *finite uniform integer time periods* $t = 0, 1, \dots, T$. In continuous time problem, the horizon time is broken up into finite periods t which can take any *real value* of the interval $t \in [0, T]$. In this work, we focus on problems with discrete finite horizon time.

16.3.2 Parameters

As in static network, each arc $a \in A$ has a capacity $c_a(t) \in \mathbb{N}$ for all $t = \{0, 1, \dots, T\}$. The capacity $c_a(t)$ defines the upper bound of flow units that can enter an arc a at each unit time t .

In dynamic network, it is possible to store the flow at nodes. This model is named *dynamic network with storage nodes*. Therefore, each node $n \in N$ has a capacity $c_n(t) \in \mathbb{N}$ which defines the maximum number of units of flow that can be held over at this node. This capacity is called *holdover capacity*. We distinguish also flow over time models without storage nodes and with infinite holdover capacity [31].

In literature, capacities of nodes and arcs are treated as constant or time-dependent attributes. Constant capacity means that each node or each arc has the same capacity for all periods $t \in [0, T]$. Time-dependent capacity implies that capacities of arcs and nodes are defined at each period $t \in [0, T]$.

In dynamic network, each arc is characterized by a transit time denoted by $\lambda_a(t) \in \mathbb{N}$. This transit time defines the time period needed to cross an arc $a = (v, w)$ departing from node v at time t , and arriving to node w at time $t' = t + \lambda_a(t)$. The transit time of a given arc depends on its physical distance and the speed of flow in this arc. Three main approaches are used to model the transit time.

- The first approach assumes that the transit time is fixed independently of the horizon time [14]. Similarly to constant capacities, constant transit time is defined once for all periods of the time horizon.
- The second approach assumes that the transit time is time-dependent [19]. This means that at each period, a constant transit time is defined in arcs. This travel time which depends on the average of speed's flow on arcs, is assumed to be constant for the travel duration on the arc at the given unit time.
- The third approach defines flow-dependent transit time. This approach takes into consideration the dependence between the transit time, the flow's rate and the flow's speed. Two main flow-dependent transit time models are proposed: inflow-dependent transit time [25] and the load-dependent transit time [24]. Inflow-dependent transit time models assume that the flow units entering an arc a at the same period t have the same speed. Hence, the flow entering an arc a with flow

rate $f_a(t)$ needs a transit time $\lambda_a(f_a(t))$ to traverse a . However, load-dependent transit time model assumes that at each period of the horizon time, the transit time $\lambda_a(l_a(t))$ of an arc a depends on load $l_a(t)$ which is the amount of flow not only entering this arc but also standing on this arc at a given time t .

16.3.3 Representation

The dynamic network $G_T = (N, A, T)$ can be presented by a discrete time-expanded network $G_T = (N_T, A_T)$ over the horizon time T as it is seen in Fig. 16.1. A time-expansion network G_T with time horizon T consists of T copies of the set of nodes N , one for each unit time t . For each movement arc $a_{M} = (v, w)$ having transit time $\lambda_a(t)$, an arc is inserted between the copy of node $v(t)$ for all $t \in \{0, \dots, T - \lambda_a(t)\}$ and the copy of node $w(t')$ for all $t' = t + \lambda_a(t)$. If we allow storage in nodes, holdover arcs are needed to connect the copy of node $v(t - 1)$ to the copy of node $v(t)$.

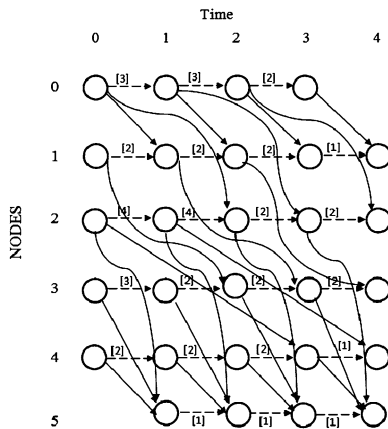


Fig. 16.1 Time-expanded network. A network with 4 copies of 5 nodes. Each copy corresponds to a node at the given unit time. *Horizontal lines* describe holdover arcs. *Other lines* are the movement arcs. Time-expanded network defines capacities for movement and holdover arcs. Only capacities of holdover arcs are illustrated

In the case of flow-dependent transit time models, Köhler et al. [25] have presented an extension of time-expansion graph which is named *fan graph*. Flow in the fan graph can use several adjacent arcs corresponding to different transit times as it is presented in Fig. 16.2.

In Fig. 16.2, where graduated horizontal lines define copies of nodes s and d at periods $t \in \{0, \dots, 5\}$. Furthermore, copies of the arc (s, d) are distinguished according to the flow rates x_1, x_2 and x_3 where $x_1 < x_2 < x_3$.

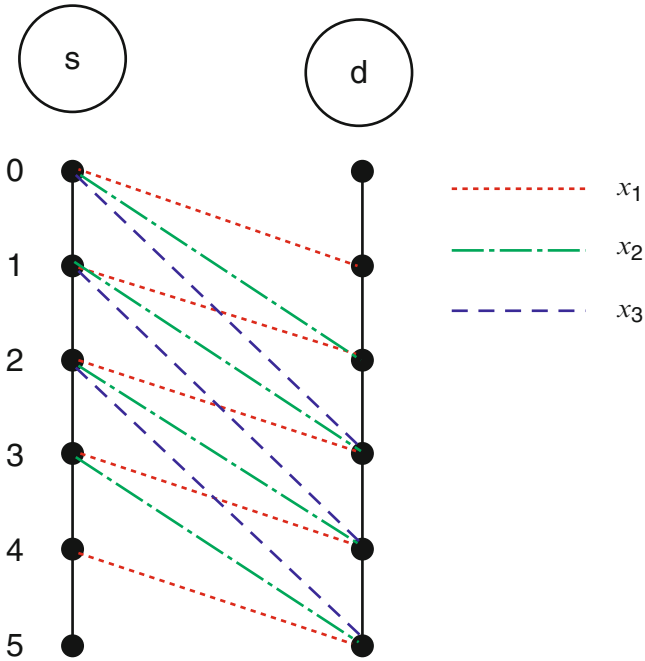


Fig. 16.2 Fan graph. This graph illustrates arcs with different capacities that connect all copies of a source node s to all copies of a destination node d . At each unit time, arcs model the different possible transit times. The capacities of arcs try to control the distribution of flow according to the transit time function. For example, if the rate of flow which is sent into the arc (s, d) is x_1 , the transit time assigned to this flow is at least $\lambda_{(s,d)}(x_1) = 1$

16.4 Flow Over Time Models

The number of flow problems that can be defined in dynamic network is obviously more important than that which can be defined in static network. Various extensions of dynamic network flow problems are treated. The variation of flow over time models is based on objective of the problem, characteristics of arcs and nodes (constant, time-dependent or flow-dependent), the type of the network (single source and sink, multiple sources and sinks or the value of demands and supplies on sources and sinks...) and models of the time horizon (discrete or continuous time). Depending on objectives of different flow over time problems, this section summarizes basic dynamic approaches proposed in literature.

16.4.1 Maximum Dynamic Flow Problem

The maximum dynamic network flow problem (MDFP) is firstly introduced by Ford and Fulkerson [14]. It tries to send the maximum flow from the source to the sink at a given horizon time T . This problem has been studied on dynamic network having constant attributes (capacities and transit time) with single source and single sink. A polynomial algorithm has been proposed to get an optimal solution. This algorithm consists on the repetition of the feasible flow along some chains of the static network from the source to the sink for every time period within T . This approach is called the *temporally repeated flow algorithm*.

Minieka [27] has modified this technique to solve MDFP in a dynamic network where arcs can be added or removed in any time period. Later, Halpern [18] has treated MDFP with time-varying capacities and MDFP without storage in nodes. Tjandra [31] has presented a pseudo polynomial algorithm for MDFP with time-varying attributes where capacities and transit time are time-dependent. Fleischer and Skutella [11] have studied the MDFP with flow-dependent transit time. Furthermore, dynamic maximum multi-commodity flow problem has been studied by Awerbuch and Leighton [1] and Hall et al. [17]. Approximate algorithms have been proposed for this problem with and without storage nodes.

16.4.2 Earliest Arrival Flow Problem

The earliest arrival flow problem (EAFP), also called universal maximum flow problem has been introduced by Gale [15]. The EAFP is an extension of the maximum dynamic flow problem. This problem consists on the maximization of the flow reaching the sink not only for the time horizon T , but also for every time $T' < T$. Therefore, the earliest arrival flow presents an additional property named the *earliest arrival property*. Minieka [27] and Wilkinson and John [32] have proposed polynomial algorithms based on the successive shortest path technique for the EAFP with constant parameters. Under the same assumptions, Hoppe and Tardos [20] have developed a polynomial $(1 + \varepsilon)$ -approximation algorithm to solve the same problem. This algorithm, named *the capacity scaling shortest augmenting path*, has been adapted later by Hamacher and Tjandra [19] to solve EAFP with time-dependent attributes. In addition, Cai and Sha [7] and Fleischer [10] have provided pseudo-polynomial algorithms to solve the same problem. In the case of flow-dependent transit time, Baumann and Köhler [3] have showed that the earliest arrival property can not be verified. This means that if we try to maximize the flow at each unit time, we can't obtain the maximum at the end of the horizon time. Hence, they proposed a relaxation which looks for the maximum flow with minimum lateness.

Furthermore, Hajek and Ogier [16] have given the first polynomial algorithm for the transshipment EAFP in the case of several sources and single sink with zero transit time. In the case of time-dependent attributes, this problem has been treated by a FPTAS approximation algorithm [11] and an exact polynomial algorithm [4].

16.4.3 Quickest Flow Problem

The quickest flow problem (QFP) asks to minimize the time needed to send a finite value of flow in source node to the sink [26]. This time is called *the network clearance time*. Burkard et al. [5] have given a polynomial algorithm for QFP with constant attributes. Also, Tjandra [31] has presented a pseudo polynomial algorithm to this problem with time-dependent attributes. Further, the QFP with flow-dependent transit time has been treated by Köhler and Skutella [24] and an approximation algorithm has been proposed. They provide a $(2 + \varepsilon)$ -approximation using temporally repeated flow technique.

The quickest transshipment problem seeks to minimize the time needed to satisfy a given supplies and demands in sources and sinks within a minimum time. Hoppe and Tardos [21] have described a polynomial algorithm to solve this problem by considering constant attributes. Fleischer [9] envisaged the quickest transshipment problem with zero transit time. He proposed an approximation algorithm to solve the problem. Further, the multi-commodity QFP has been treated with constant attributes by Fleischer and Skutella [11] and with inflow-dependent transit time by Hall et al. [17].

16.4.4 Dynamic Minimum Cost Flow Problem

The dynamic minimum cost flow problem seeks to find the flow that satisfies supplies and demands in a given horizon time such that the total cost defined on arcs or nodes is minimized. Therefore, if costs on arcs are given, then we fix the amount of flow $val(X)$ and the horizon time T , and we ask for the minimum cost dynamic flow on arcs that sends $val(X)$ amount of flow from the source to the sink over T . Hence, the minimum cost maximum dynamic flow problem tries to maximize the flow and minimize the cost of a given $val(X)$ in a given horizon time. Whereas, the minimum cost quickest flow problem seeks to minimize the cost and the time T needed to send a given flow $val(X)$. Kotnyek [26] has defined these two problems and has proved that both are NP-hard.

Klinz and Woeginger [23] considered the s - t -dynamic minimum cost flow problem with constant attributes and without waiting in nodes. Fleischer and Skutella [12] treated transshipment dynamic minimum cost flow problem with the same constraints. Approximation algorithms have been proposed for these problems. s - t -minimum cost dynamic flow problem on time-varying networks has been handled by Cai and Sha [7]. In this work, the time-varying attributes are costs, travel times and capacities on arcs as well as on nodes. Pseudo-polynomial algorithms are provided to solve this problem with waiting capacity, with unlimited waiting in nodes and with the prohibition of waiting in nodes.

16.4.5 Complexity of Dynamic Network Flow Problems

We summarize in this section the complexity of flow over time problems. In Table 16.1, three sets of flow over time problems are considered: single source and single sink flow problems, transshipment problems and multi-commodity problems [24]. We can note that for flow over time problems with constant transit time, polynomial (Poly) solutions has been proposed [14, 15]. Dynamic network flow problems with time-dependent transit time [31] are classified pseudo-polynomials (Pseudo-Poly). However, no polynomial results have been presented for transshipment problems [9], multi-commodity problems [11] and the flow over time problems with flow-dependent transit time [25]. These problems are classified as NP-hard.

Table 16.1 Complexity of flow over time problems

Dynamic problems	Single source-single sink	Transshipment	Multi-commodity
Constant transit time	Poly	Poly	Pseudo-poly
Time-dependent transit time	Pseudo-poly	NP-hard	NP-hard
Flow-dependent transit time	NP-hard	NP-hard	NP-hard

Approximation algorithms have been proposed to solve NP-hard flow over time problems. These algorithms define a guarantee on the bound of the obtained solution from the global optimum. However, in practice, these algorithms present several limits to solve problems with large size instances. Indeed, dynamic network flow models have been used to simulate real-life applications where large scale instances are needed as for evacuation and telecommunication. In these cases, decision makers need efficient solutions in reasonable time.

For this reason, we choose in this work to use metaheuristics to handle NP-hard dynamic network flow problems, and more precisely those with flow-dependent transit time. This was motivated by the ability of these techniques to solve real-life NP-hard problems.

16.5 Metaheuristics

Metaheuristics have a good potential for resolving various NP-hard problems since they use strategies to escape local solutions and to allow the search space to be explored efficiently. Unlike exact methods, there is no guarantee to find optimal or even bounded solutions.

Before implementing a metaheuristic, three main parameters have to be defined: the representation of the solution handled by algorithms, the definition of the objective function that will guide the search and the definition of the constraint handling strategy. Many classifications of metaheuristics have been proposed by Talbi [30]. In this study, we distinguish between evolutionary-based metaheuristics and blackboard-based metaheuristics.

16.5.1 Blackboard-Based Metaheuristics

Blackboard-based metaheuristics are inspired from the collective behavior of agents. Ant colony [8] and artificial bee [22] algorithms are classified as blackboard-based metaheuristics. These algorithms are based on the cooperative construction of the solution. Blackboard-based algorithms are greedy algorithms where each agent tries to construct an efficient solution by adding solution components to a partial one until a complete solution is derived. In addition, each agent contribution is done according to the shared memory and heuristic information. The shared memory keeps characteristics of the best generated solution. Heuristic information represents known information specific to the problem.

16.5.2 Evolutionary-Based Metaheuristics

Evolutionary algorithms operate on complete solutions unlike blackboard-based algorithms. They try to improve iteratively, the solution's quality. We distinguish between two classes of evolutionary algorithms: single-solution based evolutionary algorithms and population-solution based evolutionary algorithms [30].

Single-solution based evolutionary algorithms start with a solution generated randomly or using a method. A neighborhood is generally obtained by local transformation or mutation of the given solution. Furthermore, selection is performed from candidate solutions from the neighborhood to replace the current solution. Local search algorithm, tabu search and simulated annealing are considered as examples of single-solution based evolutionary algorithms.

Population-solution based evolutionary algorithms operate on a set of solutions called population. Initially, a population is generated randomly or using specific methods. At each step, individuals are selected to reproduce new offsprings using variation operators. A replacement scheme is applied to determine which individuals of the population will survive from offsprings and parents. This generation is iterated until a stationary state is reached [33].

16.6 An Evolutionary Framework for Dynamic Network Flow Problems

In this section, we provide an evolutionary framework to treat dynamic network flow problems. This framework is generic since it can be adapted to different models and problems presented in Sect. 16.4.

16.6.1 Solution Representation

A solution in a dynamic flow problem corresponds to a set of assignments of flow to each node or arc at each period. Since, we can deduce the flow on arcs if we have the flow on nodes or inversely, we can choose to represent the flow distribution either on nodes or arcs.

More formally, let $G_T = (N, A, T)$ be a dynamic network, each solution is described by the flow matrix F . Indeed, F has as dimension $|N| \times T$ entries or $|A| \times T$. Each entry F_{it} assigns to each node $i = 1 \dots |N|$ (respectively $i = 1 \dots |A|$) a number of flow X at each unit time $t = 0 \dots T$. Hence, each row r_t in F corresponds to the state of G_T at unit time t .

16.6.2 Generation of Initial Solutions

For flow over time problems, solutions have to be feasible by verifying standard constraints related to flows on network. In addition, the state of the network in a given time t depends on its state in time $t - 1$. Thus, to generate initial solutions for dynamic flow problems, we recommend the use of hybrid strategies. Indeed, initial solution should be constructed in a greedy way and flow on nodes and arcs have to be generated randomly to ensure diversification. Furthermore, these values have to satisfy basic constraints such as capacity's constraint and flow's conservation constraint.

16.6.3 Crossover Operator

The crossover operator is binary and sometimes n-ary. The role of crossover operators is to inherit some characteristics of parents to generate the offsprings. For flow over time problems, we define a crossover operator as follows.

Given two selected solutions, represented by two matrix F_1 and F_2 (parents), we have to form a new solution (offspring) F_3 . At each unit time, each node in F_1 is compared to the corresponding node in F_2 and the flow of the dominant node is put

in F_3 at the same period. We say that a node dominates another node if it contains more flow units in the case of maximization of flow. This definition can be changed according to the chosen fitness function of the problem.

After combination, the offspring F_3 have to be transformed to a feasible solution by adjusting the flow on the network to respect constraints on nodes and arcs.

16.6.4 Mutation Operator

Mutation operator is unary operator acting on single solution. This operator can be simple by flipping a cell in the current matrix corresponding to a solution, or more sophisticated by applying a local search procedure on the chosen solution.

16.7 Application to Evacuation Problem

Emergency evacuation is needed in many situations such as fire, floods, volcanoes, tsunamis and terrorism acts. This evacuation consists in the movement of people from dangerous area to safe one as quickly as possible. In order to obtain an optimal evacuation process, researchers sought for methods to find an efficient evacuation plan which maximizes the number of saved evacuees.

Different flow over time models has been used to simulate evacuation problem. We choose, to deal with dynamic maximum flow problem with flow-dependent transit time (DMFP-FDT) using a genetic algorithm based on the framework proposed in the previous section.

We consider a graph G_T with $c_{ij}(t)$, $c_i(t)$, $x_{ij}(t)$ and $l_{ij}(t)$ which are respectively the capacity of arc (i, j) , the capacity of node i , the flow value and the load of the arc (i, j) at period t . The dynamic maximum flow problem is modeled as follows [11];

Maximize

$$val(X) = \sum_{t=0}^T \sum_{(i,d) \in A} x_{id}(t) \tag{16.3}$$

Subject to

$$x_{ij}(t) \leq c_{ij}(t), \forall (i, j) \in A, t \leq T \tag{16.4}$$

$$x_i(t) \leq c_i(t), \forall i \in N, t \leq T \tag{16.5}$$

$$\lambda_{ij}(t) = \lambda_{ij}^0 * (1 + \gamma(l_{ij}(t)/c_{ij}(t))) \tag{16.6}$$

$$\sum_{(i,j) \in A} \sum_{t'} x_{ij}(t') - \sum_{(i,j) \in A} \sum_t x_{ij}(t) = x_i(t) - x_i(t-1) \tag{16.7}$$

$$t' + \lambda_{ij}(t') = t \tag{16.8}$$

Equation (16.3) describes the objective of maximization of saved evacuees. This function computes the value of flow $val(X)$ arriving to destination node at each unit time. Equations (16.4) and (16.5) represent constraints of capacity on arcs and nodes respectively. Since capacities of arcs and nodes are time-dependent, it is necessary to verify these constraints at each period. Equation (16.6) illustrates the transit time function which defines the dependence between transit time and flow on arcs. Equation (16.7) defines the conservation constraint which computes the load in nodes at each unit time. This constraint allows the storage of flow in nodes. The period t' used in Eq. (16.7) is defined by Eq. (16.8).

The transit time function in Eq. (16.6) is the *Bureau of Public Roads* function (BPR) defined by Sheffi [29]. This transit time function has been used by Baumann [2] for the earliest arrival flow problem with flow-dependent transit time. The BPR function depends on λ_{ij}^0 which is the transit time of the arc (i, j) when it is empty. $l_{ij}(t)$ is the load of the arc (i, j) at unit time t and $c_{ij}(t)$ is the capacity of the arc (i, j) at unit time t . The value γ weights the importance of crowdedness in this transit time.

16.7.1 A Case Study for Building Evacuation

We propose an evacuation planning case study for the second floor of children hospital (see Fig. 16.3) located in the city of Bab Saadoun in Tunisia. The purpose of this application is to investigate the performance of the proposed metaheuristic in evacuation networks simulated using flow over time models. We apply a genetic algorithm to find a maximum number of saved evacuees for a given period in a time-dependent environment taking into consideration the crowdedness in corridors and doors.

The plan of the hospital is transformed to static network consisting of 30 nodes represented in Fig. 16.4. Nodes correspond to rooms and corridors of the hospital and arcs corresponds to doors. This network defines multiple sources and one sink.

The sink is represented by node 30. Nodes drawn by a circle surrounded by a rectangle describes sources nodes. The network can be transformed into network with single source by adding an artificial node named super source which is connected to the set of sources. The capacity of this node will be the sum of source nodes capacities. Arcs related to this node have zero transit time.

We predict an emergency situation which need 20 min to evacuate all people from the building. We define one unit time by 1 min. Hence, the time horizon T is broken to 20 periods. Table 16.2 lists all nodes with positive initial load where the total number of evacuees is set to 965 at the first unit time.

The transit time λ_{ij}^0 and initial capacities of arcs and nodes at the first unit time are specified by an architect depending on the plan of the building. Time-dependent capacities data are defined randomly. Initial capacities of nodes are presented in Table 16.3.

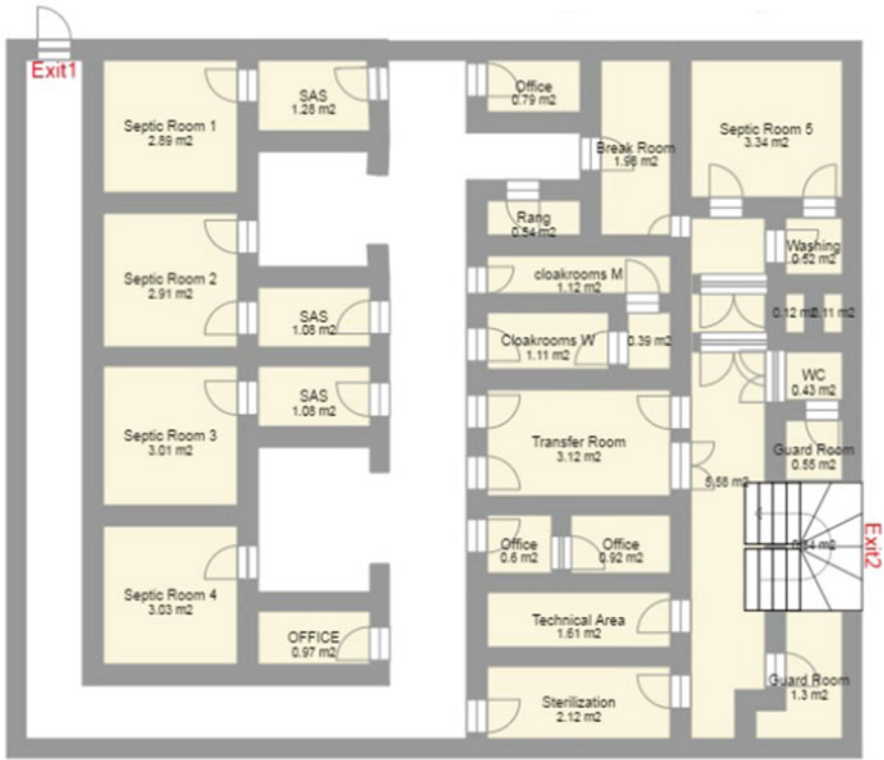


Fig. 16.3 Plan of the second floor of children hospital

16.7.2 Design of Genetic Algorithm

Genetic algorithms (GA) have a good potential to solve various dynamic NP-hard problems. We propose this GA based on a population of solution represented by the matrix of flow in nodes at each unit time. As it is seen in Sect. 16.6, we have to adapt algorithms generating initial population, crossover and mutation in order to check constraints and characteristics of flow over time problems.

16.7.2.1 Initial Population

The generation of flow in nodes can not be completely random, since the state of nodes at a given unit time depends on their state and the state of predecessor nodes in previous unit times. Hence, we choose to use a constructive random dynamic algorithm. This algorithm is based on temporally repeated flow technique [14] using random units of flow and flow-dependent transit time.

We propose the Operator 1 to generate initial population. This heuristic uses the dynamic residual network defined by Tjandra [31]. At each unit time, a residual network is constructed. From each node n with positive flow $x_n > 0$ a random flow

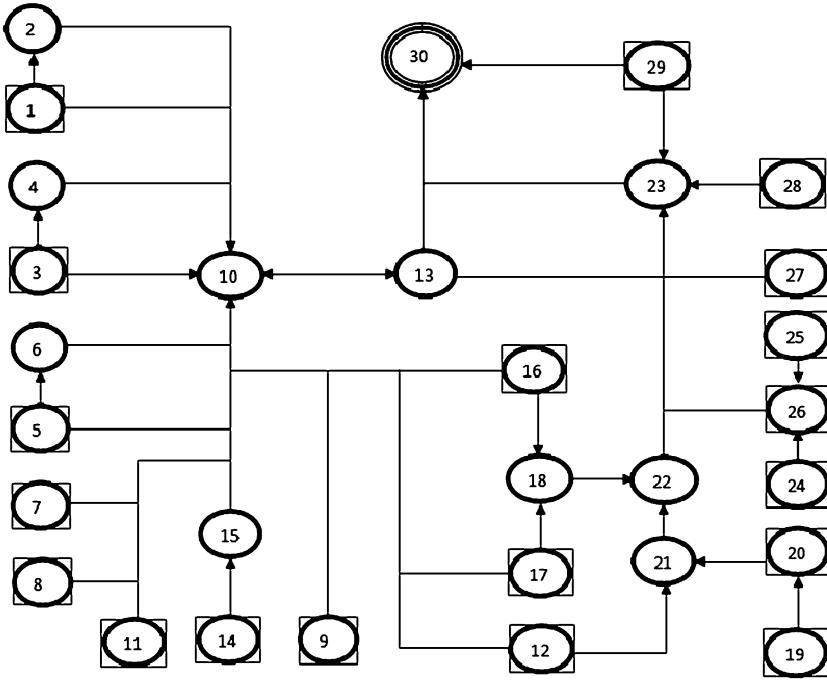


Fig. 16.4 Network representation of the second floor of children hospital

Table 16.2 Initial load of nodes

Node	1	3	5	7	8	9	11	12	14	Node	16	17	19	20	24	25	26	27	28	29
Initial load	100	100	100	100	30	30	30	100	15	Initial load	60	60	6	9	21	30	21	45	33	75

Table 16.3 Initial nodes’s capacities

Node	1	2	3	4	5	6	7	8	9	
Initial capacity	120	60	120	60	120	60	120	45	45	
Node	10	11	12	13	14	15	16	17	18	19
Initial capacity	60	45	120	50	25	35	70	70	60	15
Node	20	21	22	23	24	25	26	27	28	29
Initial capacity	15	10	10	20	27	40	30	50	40	100

f is generated at each outgoing arc a with capacity c_a . Therefore the transit time $\lambda_a(f)$ is computed following the formula (16.6).

16.7.2.2 Crossover and Mutation

Since the objective of this study consists on maximization of flows in destination node, offspring should inherit nodes with maximum flow from parents. Hence, crossover operator consists on assigning to offspring the maximum load from parents’ solutions for each node at a given period. We choose to apply crossover for the set of nodes at the first five periods.

Operator 1 Algorithm for generating initial population

```

Set dynamic parameters of the graph
for all  $\theta \in T$  do
  for all  $n \in N$  with flow not null do
    for all  $a \in \zeta^-(n)$  do
      Generate Random flow  $f \in [0, x_n]$  in  $a$ 
      if  $f > c_a(t)$  then
         $f = c_a(t)$ 
      end if
      Compute the transit time  $\lambda_a(f)$ 
      Assign  $f$  to destination node of arc  $a$  for period  $\theta + \lambda_a(\theta)$ 
    end for
  end for
end for

```

Other periods of offspring are constructed using the algorithm proposed for the generation of initial population during the mutation. At each generation, offsprings are evaluated and best solutions from parents and offspring are sent to next generation.

16.7.3 Results Analysis

We implemented the GA using Paradiseo framework [30] based on C++ language. These experiments were conducted on a centrino duo 1.66 GHz laptop. Experiments have been performed to maximize the number of people that can be saved when we assume that transit time is flow-dependent.

Table 16.4 Representation of the best solution obtained by the proposed GA

Periods	T_0	T_1	T_2	...	T_{20}
Node 0	965	572	316	...	0
Node 1	0	7	17	...	0
Node 2	0	1	1	...	0
Node 3	0	42	67	...	55
...
Node 30	0	47	71	...	771

Table 16.4 defines the solution representation of the solution provided by our adaptation of GA where rows correspond to nodes and columns to periods. We remind that this solution describes flow in nodes at each period. Hence, the GA procedure can generate at each iteration several evacuation plans and tries to improve the evacuation process using genetic operators.

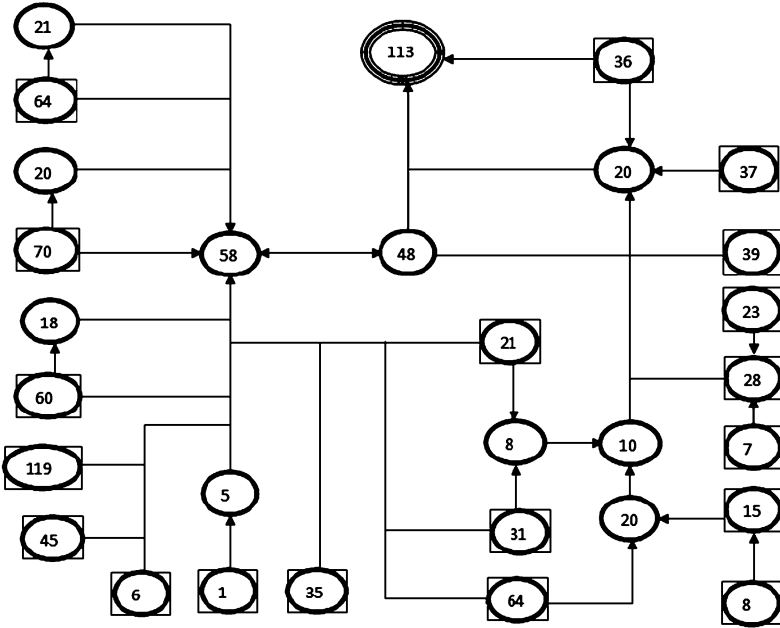


Fig. 16.5 Distribution of evacuees at period 5 in random plan from initial population

Indeed, in Fig. 16.5, we show the distribution of evacuees after 5 min in the initial solution (plan) which is constructed randomly. This distribution demonstrates that after five periods, source nodes are still crowded such as node 1 which contains 64 persons. However, in this distribution, 113 persons which reach destination node are saved.

Figure 16.6 describes evacuation plan at the end of evacuation period (at period 20). In this figure, the distribution of evacuees in the building after 20 min shows that an important number of evacuees are saved (771 persons reach destination node).

Therefore, this figure shows that several nodes such as node 13 and node 10 still contain people. This proves that crowdedness in these nodes increases the transit time of these people. This phenomena can be explained by the fact that these nodes have several predecessors and only one successor. Hence, we recommend the adding of other doors to these rooms. In addition, we believe that it is crucial to prevent crowdedness by installing cameras in these locations. Therefore, this solution may be improved by the increase of the time needed to evacuate people (evacuation time T).

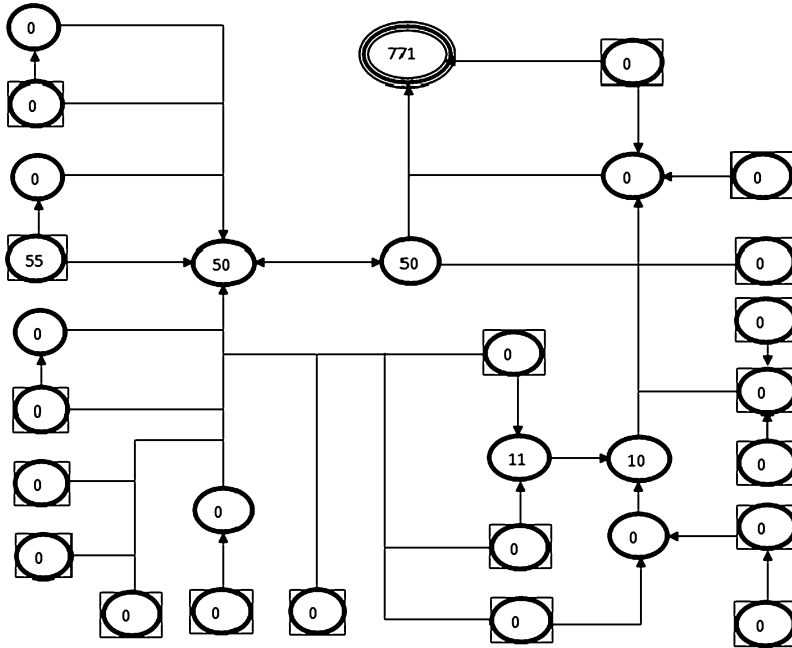


Fig. 16.6 Distribution of evacuees at period 20 in the best plan

16.8 Conclusion

Our survey of the state of the art of network flow problems states clearly that flow over time problems could adequately model several real applications. Dynamic network flow models append the time factor to classical static models. Dynamic network flow models differ according to three parameters: the type of the network, the objective of the problem and the attributes’s proprieties (transit time and capacities). These variants do not exhibit the same complexity. We distinguish polynomial, pseudo-polynomial and NP-hard problems.

The contribution of this chapter is the definition of a metaheuristic framework for NP-hard flow over time problems. A specific case study of dynamic flow problem is treated, precisely the evacuation problem from a building. Therefore, we have supposed that the dynamic maximum flow model with flow-dependent transit time could handle the dynamic property and the crowdedness on nodes and arcs. We choose the genetic algorithm as a population-based evolutionary method to treat this NP-hard problem.

The GA enables the search to reach evacuation plans that could simulate real evacuation process. In addition this algorithm enabled the location of critical areas that should improve their security by implementing cameras or adding exits. The

proposed evolutionary framework can also be applied by testing single-solution evolutionary approaches such as tabu search or simulated annealing. Furthermore, other operators should be tested and parameters could be adjusted accordingly.

References

1. B. Awerbuch, T. Leighton, Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks, in *STOC '94 Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, 1994
2. N. Baumann, Evacuation by earliest arrival flows. PhD thesis, University Dortmund, Dortmund, 2006
3. N. Baumann, E. Köhler, Approximating earliest arrival flows with flow-dependent transit times. *Discr. Appl. Math.* **155**, 161–171 (2007)
4. N. Baumann, M. Skutella, Earliest arrival flows with multiple sources. *Math. Oper. Res.* **34**, 499–512 (2009)
5. R. Burkard, K. Dlaska, B. Klinz, The quickest flow problem. *ZOR Methods Models Oper Res.* **37**, 31–58 (1993)
6. R. Busacker, P. Gowen, A procedure for determining a family of minimal-cost network flow patterns. Technical report, Johns Hopkins University (1960)
7. X. Cai, D. Sha, Time-varying universal maximum flow problems. *Math. Comput. Modell.* **33**, 407–430 (2001)
8. M. Dorigo, T. Stützle, Ant colony optimization: overview and recent advances, in *Handbook of metaheuristics*, vol. 164, 2010, pp. 227–263
9. L.K. Fleischer, Faster algorithms for the quickest transshipment problem. *SIAM J. Optim.* **12**, 18–35 (2001)
10. L.K. Fleischer, Universally maximum flow with piecewise constant capacity functions. *Networks* **38**, 115–125 (2001)
11. L. Fleischer, M. Skutella, The quickest multicommodity flow problem, in *Integer Programming and Combinatorial Optimization*, vol. 2337, 2002, pp. 36–53
12. L. Fleischer, M. Skutella, Minimum cost flows over time without intermediate storage, in *SODA'03 Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003, pp. 66–75
13. L. Ford, D. Fulkerson, Maximal flow through a network. *Class. Pap. Combin.* **8**, 243–248 (1956)
14. L. Ford, D. Fulkerson, Constructing maximal dynamic flows from static flows. *Oper. Res.* **6**, 419–433 (1958)
15. L. Gale, Transient flows in networks. *Mich. Math. J.* **6**, 59–63 (1959)
16. B. Hajek, R.G. Ogier, Optimal dynamic routing in communication networks with continuous traffic. *Networks* **14**, 457–487 (1984)
17. A. Hall, S. Hippler, M. Skutella, Multicommodity flows over time: efficient algorithms and complexity. *Theor. Comput. Sci.* **379**, 387–404 (2007)
18. J. Halpern, A generalized dynamic network flows problem. *Networks* **9**, 133–167 (1979)
19. H. Hamacher, S. Tjandra, Mathematical modeling of evacuation problems: a state of the art, in *Pedestrian and Evacuation Dynamics*, vol. 19, 2002, pp. 227–266
20. B. Hoppe, E. Tardos, Polynomial time algorithms for some evacuation problems, in *SODA '94 Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1994, pp. 433–441
21. B. Hoppe, E. Tardos, The quickest transshipment problem. *Math. Oper. Res.* **25**, 36–62 (2000)
22. D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony algorithm and applications. *Artif. Intell. Rev.* **37**, 1–37 (2012)

23. B. Klinz, G. Woeginger, Minimum cost dynamic flows: the series-parallel case, in *Integer Programming and Combinatorial Optimization*, vol. 920 (1995), pp. 329–343
24. E. Köhler, M. Skutella, Flows over time with load-dependent transit times. *SIAM J. Optim.* **15**, 1185–1202 (2005)
25. E. Köhler, K. Langkau, M. Skutella, Time-expanded graphs for flow-dependent transit times, in *Algorithms-ESA*, vol. 2461, 2002, pp. 599–611
26. B. Kotnyek, An annotated overview of dynamic network flows. Technical report, National Institute of Research on Computer and Automata (2003)
27. E. Minieka, Dynamic network flows with arc changes. *Networks* **4**, 255–265 (1974)
28. J. Orlin, Minimum convex cost dynamic network flows. *Math. Oper. Res.* **9**, 190–207 (1984)
29. Y. Sheffi, *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods* (Prentice-Hall, Englewood Cliffs, 1985)
30. E.G. Talbi, *Metaheuristics: From Design to Implementation* (Wiley, New York, 2009)
31. S. Tjandra, Dynamic network optimization with application to the evacuation problem. PhD thesis, University Kaiserslautern, Kaiserslautern, 2003
32. W.L. Wilkinson, J. John, An algorithm for universal maximal dynamic flows in a network. *Oper. Res.* **19**, 1602–1612 (1971)
33. Y.G. Woldesenbet, Dynamic evolutionary algorithm with variable relocation. *IEEE Trans. Evol. Comput.* **13**, 500–513 (2009)

Chapter 17

A Greedy Randomized Adaptive Search for the Surveillance Patrol Vehicle Routing Problem

Simona Mancini

Abstract In this chapter a new rich vehicle routing problem is introduced, the Surveillance Patrol Vehicle Routing Problem (SPVRP). This problem came out from a real need of a surveillance company to create fairer routing plans for its security patrols. The problem consist into routing a set of patrols in order to visit a set of checkpoints. Each checkpoint requires one or more visits, each one of which, to be performed within a fixed time window. A minimum time spacing between two consecutive visits should be observed. The goal is to minimize cost while minimizing, at the same time, time windows and minimum spacing constraints violations. In order to avoid repetitiveness in the routes and to provide more unpredictable routing plans, the company looks for a pool of sensibly different high quality solutions from which, each night they can choose the routing plan to be followed. To address this problem a Greedy Randomized Adaptive Search algorithm (GRASP), is used to provide good solutions and a further GRASP algorithm is used to generate pools of good solutions. The quality of a pool is measured both in terms of averaged quality of the solutions in the pools and in terms of diversity among each others. Experimental tests on real instances are reported.

Keywords Rich Vehicle Routing • Multiple Time Windows • Heterogenous fleet • Greedy Randomized Adaptive Search

17.1 Introduction

In the last decades the request of private security and surveillance services constantly grows. Customers became more and more exigent, and their requests may be completely different among each others. They may require single or multiple visits per night, with different time windows which may be completely disjointed, overlapped or partially overlapped. For instance, a customer may require three visits,

S. Mancini (✉)
Politecnico di Torino, 10129 Torino, Italy
e-mail: simona.mancini@polito.it

in which one must be achieved before midnight, one between midnight and 2 o'clock in the morning and one after 4 o'clock, while another one may request three visits in which the only rule is that they must be carried out all before 4 o'clock. The case of partially overlapped time windows occurs when, for instance, one visit is required before 2 o'clock and the other two after midnight. The location to be visited, from now on called checkpoints, may be completely different both in typology and in size. They may vary from a large shopping center, which requires several minutes to be checked, to a cash dispenser. Furthermore, the same customer may require different types of visits during the night. For instance, a shopping center may require a longer visits in the first hours of the night (just after closing time) and short checks during the night. Moreover, a minimum spacing time between two consecutive visits must be ensured. In fact, if a checkpoint must be visited once before 2 o'clock and once after 2 o'clock, it is not fair to visit it at 1:55 a.m. and at 2:05 a.m. The difficulty of a task depends on the checkpoint location (checking a cash dispenser near the train station may be much more dangerous than one located in an uptown district) and typology. Patrols may be grouped by level of expertness, and only certain group of patrols may visit the most dangerous checkpoints. The fixed cost associated to the use of a specific patrol is proportional to its level of expertness.

A surveillance company aims to reduce costs, given by fixed cost for patrol usage plus kilometric cost multiplied for the total covered distance, while at the same time minimizing the number of time windows and minimum spacing constraints violated. This because they aim to be more competitive on the mark and to be able to offer a high quality service level at a competitive price. The final goal of the company is not only to schedule a fair routing plan at the minimum cost, but also to define a set of potential high quality plans from which to choose, each day, a different plan to be followed. This request came out from the need to avoid to repeat always the same tours, for security reasons. Indeed, if criminals know exactly at what time a patrol visit a checkpoint, they may easily evade surveillance. More unpredictable is the visits schedule, more difficult is to elude surveillance, with a consequent increase of the public safety level and a reduction of committed crimes. For this reason a second optimization problem may be defined, consisting into choosing a subset of Q solutions within a set of N feasible solutions, in order to minimize cost, time windows and minimum spacing constraints violation and repetitiveness. More in detail, two repetitiveness indexes can be defined, the first one, deals with visit times at checkpoints and works as follows. If, a checkpoint is visited within the same timeslot in more than $r\%$ of the solutions belonging to the subset, a visit time repetitiveness penalty is added. A similar procedure is applied for sequence repetitiveness. In fact, if a checkpoint is visited immediately after the same checkpoint in more than $q\%$ of the solutions in the subset, a sequence penalty is added. The value of r and q are parameters of the algorithm. Timeslots are defined as very small consecutive disjointed time intervals (i.e. 10 min).

17.2 Literature Review

Considering all the requirements and features described in the previous section, planning and scheduling routes for surveillance patrols becomes a very challenging problem, much more complex than the classical extensions of the vehicle routing problem (VRP), broadly studied in the literature. In particular, the SPVRP can be described as a multi-objectives version of the VRP with Multiple Time Windows and Heterogeneous Fleet with additional minimum spacing constraints. This problem combines different features which have never been addressed together in the literature, but which have been broadly studied separately. Multi-objectives vehicle routing problems frequently occur in real-life applications. The most studied problem in this field is the VRP with Workload Balance, introduced by Lee and Ueng [13] in which the goal is twofold: minimize the total traveled distance (or more generally the total travel cost) and balance the workload of the drivers. In [16] the goal is to minimize the routing cost trying to maximize customers satisfaction, which can be achieved minimizing time windows violations, which is also one of the goal of the SPVRP. Many different objectives may be found in practical applications such as the Schoolbus Routing Planning, addressed in [2], in which four objectives are considered: the minimization of the total route length, the minimization of the total student walking distance, the fair distribution of the load (i.e., the number of students transported), and the fair division between the buses of the total distance traveled. Other objectives addressed in the literature are the minimization of the longest route, as specified in an application in urban trash collection [12], and the balancing of time spent on the bus by the scholars, in a Rural Schoolbus Routing Planning [3]. Multiple Objectives optimization is very frequent in Hazardous Material Transportation, where the objectives addressed are the routing cost minimization and the risk minimization [9, 19]. For a complete survey on multi-objectives VRPs the reader may refer to [11].

Vehicle-routing problems with time windows (VRPTW) have been broadly addressed in the literature. Two different types of Time Windows may be defined: Hard Time Windows, which must be respected, Soft Time Windows which may be violated paying a penalty in the objective function. The Vehicle Routing Problem with Hard Time Windows (VRPHTW), has been extensively studied and hundreds of solution approaches, both exact and heuristics have been proposed. For a complete survey on this subject the reader may refer to [17]. Soft time Windows have received limited attention respect to their hard counterpart, even if they are more frequently addressed in real-life applications. A survey on papers dealing with Vehicle Routing Problems with Soft Time Windows is reported in [14]. In the SPVRP, Soft Time Windows are considered; in fact, the minimization of TW violations is one of the objective of the problem.

The Heterogeneous Fleet Vehicle Routing Problem (HVRP), is an extension of the classical VRP in which customers are served by a heterogeneous fleet of vehicles with various capacities, fixed usage costs, and variable cost per distance unit. An extensive literature review on HVRP is reported in [1].

Despite its high relevance in real-life applications, the spacing constraints it is not often studied in the literature. In fact, in many real applications, such as in the SPVRP, it is important not only the period within which the visit is carried out but also the moment, within that period, in which it is actually performed.

The need of ensuring diversity among routing plans across different days in the time horizon, which is of crucial importance in the SPVRP, has never been addressed before in literature, while a similar but specular issue is treated in the Consistent Vehicle Routing Problem (ConVRP), introduced by Groer et al. [10], in which the same driver must visit the same customers at roughly the same time on each day that these customers need service. In SPVRP, diversity is treated as a soft constraint, adding a penalty in the objective function if it is violated, while consistency in ConVRP is treated as a hard constraint.

17.3 The Surveillance Patrols Vehicle Routing Problem

In this section a formal description of the Surveillance Patrols Vehicle Routing Problem (SPVRP) is reported. The problem consist into routing a set of patrols P in order to visit a set of checkpoints C . Each checkpoint c may require a different number of visits $|V_c|$ (being V_c the set of visits to checkpoint c) each one to be exploited within a fixed time window. The routes scheduling is done across a single period. A minimum time spacing, δ , between two consecutive visits should be observed for all the checkpoints. Each visit to a checkpoint is characterized by a service time, s_{cv} , representing the time to be spent at checkpoint c during visit v (this time may vary from 1–2 min to 15–20 min depending on the service required). Each checkpoint is associated with a level of difficulty d_c , basing on its degree of danger. Checkpoints located in residential zones generally have a low dangerousness, due to the low number of crimes committed in the zone, while checkpoints located in slum neighborhood or near the railway station or places with a high affluence of people like stadiums, are potentially much more dangerous and must be controlled only by expert patrols. Each patrol p is associated with an expertness level, l_p such as a patrol p may be assigned to a checkpoint c only if it holds the minimum experience level required by c , i.e. only if $l_p \geq d_c$. A checkpoint c can be visited by different compatible patrols. Obviously more expert patrols cost is higher than less experted ones; more precisely a fixed cost, K_p is associated to each patrol and it is activated only if the patrol is scheduled in the plan. Patrols are grouped in classes of expertness and fixed costs are homogeneous within each class of experience. This means that, if two patrols, p_1 and p_2 , hold the same level of expertness, i.e. if $l_{p_1} = l_{p_2}$, then $K_{p_1} = K_{p_2}$. A limited number of patrols for each class is available. Each route start from the company station and must come back to the station before the end of the working period (generally patrols work in the period 10 p.m.–6 a.m.). A unitary distance cost v_d , equal for all the patrols, and a unitary time cost, v_{tp} varying among classes of patrols, are defined. The objectives addressed in the optimization process are the following:

- Minimization of a generalized cost function composed by
 - Patrols activation costs
 - Distance costs: equal to the total traveled distance multiplied by unitary distance cost
 - Time costs: equal as the sum over all the patrols of total time spent out of the station multiplied by unitary patrol time cost
- Minimization of minimum spacing constraints violations
- Minimization of time windows violations

A further optimization problem consists into determine the best pool of Q solutions according to the following criteria:

- Minimization of averaged generalized cost of the solutions within the pool
- Minimization of minimum spacing constraints violations
- Minimization of time windows violations
- Minimization of visit time repetitiveness
- Minimization of visits sequences repetitiveness

The visit time repetitiveness is defined as follows. If, a checkpoint c is visited within the same timeslot in more than $r\%$ of the solutions belonging to the subset, a visit time repetitiveness penalty is added. A similar procedure is applied for sequence repetitiveness. In fact, if a checkpoint is visited immediately after the same checkpoint in more than $q\%$ of the solutions in the subset, a sequence penalty is added. Timeslots are defined as very small consecutive disjointed time intervals (i.e. 10 min). The greater the value of r and q , the lower is the minimum degree of diversity requested.

17.4 A GRASP for the Surveillance Patrol Vehicle Routing Problem

The greedy randomized adaptive search procedure (GRASP) has been broadly applied to solve combinatorial optimization problems [8]. At each iteration of the GRASP a feasible solution is constructed by a greedy randomized algorithm and is improved through a Local Search procedure [5, 6, 7]. Greedy randomized algorithms are based on the same principle guiding pure greedy algorithms. However, they make use randomization to build different solutions at different runs. At each iteration, the set of candidate elements is formed by all elements that can be incorporated into the partial solution under construction without destroying feasibility. As before, the selection of the next element is determined by the evaluation of all candidate elements according to a greedy evaluation function. The evaluation of the elements by this function leads to the creation of a restricted candidate list (RCL)

formed by the best elements, i.e. those whose incorporation into the current partial solution results in the smallest incremental costs. In the first version of GRASP presented by Feo and Resende [5], the element to be incorporated into the partial solution is randomly selected from those in the RCL, but in later works, [4, 15] a fitness function is computed for all the elements and their probability to be chosen to be inserted in the solution is proportional to their fitness. Once the selected element has been incorporated into the partial solution, the set of candidate elements is updated and the incremental costs are reevaluated. Greedy randomized algorithms are extremely useful in case in which it is necessary to create several feasible good quality solutions. Since in this problem we are looking for a pool of good quality solutions, and not only for the best solution, GRASP fit well for this purpose.

The GRASP proposed for the SPVRP works as follows. At each iteration of the algorithm, a greedy solution is constructed. At each step of the construction phase, next checkpoint to be visited is randomly drawn according to distance based probability, i.e. the probability of each checkpoint to be chosen as next, is based on the inverse of the distance of the checkpoint and the last visited one (or, when the first customer of a route must be selected, based on the inverse of the distance of the checkpoint and the station). If a checkpoint has been already visited by the same route or by another one, during the current time window, or within a fixed minimum spacing time, its probability to be chosen is put equal to zero. For instance, assume that checkpoint c must be visited once during the time window 12 p.m.–2 a.m. and once during the time window 2 a.m.–4 a.m and the minimum spacing time between visits has been fixed equal to 60 min; if c has been visited once at 1:30 a.m. and if inserted in the current route at the current point it would be visited at 2:10 a.m. its probability to be chosen at this point is forced to zero, because, even if the two visits would respects time windows constraints, the minimum required spacing between consecutive visits would not be respected. A route is closed when no more available customers may be inserted in or when maximum duration is reached. We consider a *maximum duration* equal to the 80% of the actual maximum duration in order to leave enough room for adding other checkpoints to the routes during the local search phase. After a route is closed, if there are still unrouted customers, another route is created according to the same procedure. After this procedure ends, the results is a completed feasible routing plan, in which all the minimum spacing constraints and all the time windows are respected. From a scheduling point of view the solution may be still unfeasible for two reasons: (a) the number of created routes is larger than the number of patrols available, (b) the number of routes requesting at least a level of expertness l^* is larger than the number of patrols holding a level higher or equal to l^* . At this point patrols are assigned to routes according to the following procedure, named *Assign Patrols* (AP).

Operator 1 Assign Patrols

```

for all the routes  $\rho \in \mathfrak{R}$  (where  $\mathfrak{R}$  is the set of routes in the analyzed solution) do
  assign  $\rho$  to an available patrol  $p$  with lowest expertness level  $l_p$  such that  $l_p > d_c \forall c \in \rho$ 
  if No more patrols holding at least the minimum level required to cover route  $\rho$  then
    Assign  $\rho$  to a patrol  $p$  with the highest level available
    Mark route  $\rho$  as infeasible
  end if
end for
Set the infeasibility degree of the solution equal to the number of infeasible routes

```

At this point of the procedure a local search phase, described in Sect. 17.4.1, is applied. If, after the local search, the obtained solution is still unfeasible, a Feasibility Search procedure, describe in Sect. 17.4.2, is applied, and if also this procedure is not able to recover from infeasibility, we discard the solution and pass to the next iteration of the GRASP. A pseudocode of the algorithm is reported in the following:

Operator 2 A GRASP for the SVRP

```

repeat
  while there are still unrouted customers do
    open a new route
    apply Calculate Fitness (Station,  $T_0$ )
    select a checkpoint  $c^*$  to be inserted in the route
    repeat
      apply Calculate Fitness ( $c^*$ , Time in which visits to  $c^*$  ends)
      select a checkpoint  $c^*$  to be inserted in the route
    until no more checkpoints may be inserted in the route
    end while
    apply Assign Patrols
    apply Local Search
    if the solution is infeasible then
      apply Feasibility Search
    end if
  until the maximum number of GRASP iterations has been reached

```

The procedure *Calculate Fitness* takes, as input parameter, the last visited node n in the route, which can be either a checkpoint or the station (if the selected route is empty), and the time in which the visit to that node has been completed, t . (If the analyzed node is the station t is considered equal to zero). The pseudocode of the procedure is reported in Operator 3.

Operator 3 Calculate Fitness(n,t)

```

for all the checkpoints  $c$  for which at least one more visit must be scheduled do
  if  $t$ +travel time between  $n$  and  $c$  is such that arrival time in  $c$  allows to not violate time
  windows constraints nor minimum spacing constraints then
    set fitness( $c$ )=distance( $n,c$ )
  else
    set fitness( $c$ )=0
  end if
end for

```

17.4.1 Local Search

The local search phase is composed by two different operators applied sequentially:

- **Minimize Routes (MR)**: which tries to empty routes containing less than γ checkpoints, relocating the removed checkpoints in another route at the minimum relocation cost.
- **Node Exchange (NE)**: which attempts to exchange each checkpoints with other checkpoints which are located within a small radius σ and which are and served in the same time-slot. (1 h length disjointed time slots).

The MR procedure aims to reduce the number of routes in the solution, and it also helps to recover from infeasibility, even if it cannot guarantee that a feasible solution is reached. In fact, each time we empty one route, we release the assignment of one patrol that could be available to be assigned to another route which needs a more expert patrol respect whom to which it has been assigned. NE is a very fast procedure because it performs a granular exploration of the neighborhood, i.e. limit the search on promising moves trying to avoid time windows and minimum spacing constraints violations. The efficiency and effectiveness of granular neighborhoods exploration has been proved in [18]. After each iteration of the local search, the AP procedure is applied. The acceptance criteria for new explored solutions is composed by two criteria applied in a hierarchical order:

- **Criteria 1**: if a solution s' has a degree of infeasibility lower than that of the current best solution, s' becomes the current best solution (whichever the values of the other objectives)
- **Criteria 2**: if the solution s' has the same degree of infeasibility of the current solution, and the score of s' is lower than that of the current best solution, s' becomes the current best solution

The score of a solution is computed as the weighted sum of generalized cost, minimum spacing violations and time windows violations. If s' has a degree of infeasibility greater than that of the current best solution it is discarded.

17.4.2 Feasibility Search

If at the end of the local search phase the solution is still infeasible, a feasibility search procedure is applied. For each infeasible route we try to remove all the checkpoints holding a difficulty level $d_c \geq l_p$, where p is the patrol currently assigned to the route, and to relocate them in another route, which is associated with a patrol l' holding a level $l'_p \geq d_c$, at the minimum solution score increment. If, even this procedure is not able to reach complete feasibility, the solution is discarded and we pass to another iteration of the GRASP algorithm.

The three components of the GRASP, initial construction, local search and feasibility search, are devoted to different goals. In fact, in the initial construction we aim to create good quality solutions without looking explicitly at the patrol assignment feasibility. In the local search we try to recover from infeasibility as a primary goal and at the same time we try to improve solution quality. In the last phase, the feasibility search is completely oriented to the recovery from infeasibility. This alternation of goals among the three phases allows the algorithm to reach good solutions, which could be not reached with standard local search algorithms, passing through infeasible regions.

17.5 A GRASP for Solutions Pools Selection

The final goal of this problem is to provide a pool of Q good quality solutions ensuring the highest degree of diversity among each others, chosen among the N solution provided by the GRASP described in the previous section. Diversity is ensured trying to avoid sequences repetitiveness and customers visit time repetitiveness. A sequences repetitiveness penalty is added if a sequence of two consecutive checkpoints is present in more than $q\%$ of the solutions while customers visit time repetitiveness is added if a checkpoint is visited more than $r\%$ within the same time interval. The time horizon is divided in consecutive disjointed very small intervals (i.e. 10 min). At each iteration of the GRASP a new pool of solutions is constructed. The fitness of each solution is computed as a weighted sum of its score and the repetitiveness penalty that occurs if that solution is added at the current pool. This means that fitness functions are dynamic and depends on the solutions already belonging to the group. At each step, a solution is drawn according to probability inversely proportional to solution fitness. The procedure ends when Ng groups, each one composed by Q solutions, are constructed. At this point, a domination analysis of the groups set is carried out. More in details, a group g_1 is dominated by a group g_2 if g_2 is better than or equal to g_1 according to each criteria and is strictly better than g_1 in at least one criteria. The list of criteria considered is:

- Averaged generalized cost of the solutions within the pool
- Averaged Spacing constraints violations
- Averaged Time windows violations

- Visit time repetitiveness
- Visits sequences repetitiveness

A set of non dominated solutions is then reported by the algorithm.

17.6 Computational Tests

In this section are reported computational results obtained on a real instance provided by a surveillance company operating in the north of Italy. The features of this instances are reported in the following:

- **Number of checkpoints:** 125 (28 of level 3, 80 of level 2 and 17 of level 1)
- **Number of visits per checkpoint:** varying from 1 to 4
- **Total number of visits to be scheduled:** 290
- **Number of patrols:** 30 (10 of level 3, 15 of level 2 and 5 of level 1)
- **Service time:** varying from 2 to 15 min
- **Time Horizon:** 8 h (from 10 p.m to 6 a.m)
- **Minimum Spacing:** 30 min

while the parameters of the algorithm are:

- **Number of solutions generated by the GRASP (N):** 100
- **Number of solutions per pool (Q):** 20
- **Number of pools generated:** 30
- **Visit time repetitiveness percentage (r):** 20
- **Sequences repetitiveness percentage (q):** 40
- **Threshold on the number of checkpoints to determine if a route should be destroyed (γ):** 5

The value of the five objectives: generalized cost, averaged number of minimum spacing violations, averaged number of tw violations, visit time repetitiveness penalties, sequences repetitiveness penalties, for each pool, is reported in Table 17.1. At each iteration of the GRASP we obtain a different solution and the value of each objective is quite homogeneous among all the provided solutions. This is a strength point of the algorithm because it means that the GRASP was able to broadly explore different areas of the solutions space, and that was able to find several good solutions. Non dominated pools are underlined in the table. What we can note is that the averaged generalized cost is equal to 8293.97, while the best non dominated pool in terms of generalized cost presents an averaged generalized costs of 8319, which is 0.3% higher. This increment of cost is the price we have to pay to ensure diversity. The fact that this price is so small means that the GRASP was able to find good solutions with a good diversity degree among each others. Computational times are very small, even for large size instances, since the whole procedure took less than 1 s to be carried out. The proposed pools of solutions avoid visit time repetitions and only one sequence of two customers is repeated in more than 40% of the solutions, that

means that the diversity degree of the solutions within the pool is very high. On average, only 8 time windows over 290 and only 3 minimum spacing constraints over 165 are violated, which is a relevant result in such a constrained problem.

Table 17.1 Objectives value for the pools of solutions

Pool	Generalized cost	MS violations	TW violations	VT repetitiveness	S repetitiveness
1	8437	3	8	0	2
2	8463	3	7	0	2
3	8371	3	8	0	2
4	8447	3	8	0	6
5	8549	3	8	0	2
6	8519	3	9	0	4
7	8583	3	7	0	2
8	8595	3	8	0	2
9	8547	3	8	1	1
10	8484	3	8	1	3
11	8497	3	8	1	2
12	8502	3	7	0	3
13	8428	3	7	0	3
14	8456	3	7	1	2
15	8437	3	7	0	1
16	8420	3	8	0	2
17	8428	3	8	0	2
18	8470	3	9	0	2
19	8545	3	8	2	3
20	8556	3	8	0	3
21	8539	3	8	0	2
22	8553	2	9	1	1
23	8618	3	8	1	3
24	8511	3	8	0	2
25	8522	3	7	0	2
26	8492	3	7	0	3
27	8430	3	7	0	2
28	8426	3	7	1	3
29	8445	3	7	0	3
30	8319	3	9	0	4

17.7 Conclusions and Future Developments

In this chapter a new rich vehicle routing problem has been introduced and formalized, the Surveillance Patrol Vehicle Routing Problem (SPVRP). This problem came out from a real need of a surveillance company to create fairer routing plans for its security patrols. The final goal is to obtain a pool of good solutions, sufficiently diverse among each others, from which to chose a different routing and scheduling plan each day, in order to avoid predictability of the patrol itineraries. The fairness of

the solution is measured according to different criteria, regarding both generalized costs (comprehensive of time, distance and patrols salary based cost) and service quality level. A Greedy Randomized Adaptive Search (GRASP) has been developed to identify several good solutions and a further GRASP is used to create pools of good solutions diverse among each others. Finally pools are evaluated and compared among each others according to different criteria based on generalized costs, service quality and diversity, in order to find non dominated pools. The algorithm has been tested on real cases instances and show strong improvements in the solution quality respect to the routing plans adopted in the reality. The proposed algorithm is able to widely explore the solution space and to provide several not similar good solutions. As further development, it could be extended to other problems concerning similar issues, such as the Consistent Vehicle Routing Problem (ConVRP), in which the goal of the problem is to ensure similarity instead of diversity. This GRASP, given its ability of finding feasible solutions, could be applied also on highly constrained routing and scheduling problems in which is difficult even to find a feasible solution. Furthermore, it could be useful to provide high quality initial populations, with a high degree of diversity, to initialize evolutionary and population based algorithms.

References

1. R. Baldacci, M. Battarra, D. Vigo, Routing a heterogeneous fleet of vehicles, in *The Vehicle Routing Problem: Latest Advances and New Challenges*, ed. by B.L. Golden, S. Raghavan, E.A. Wasil (Springer, Heidelberg, 2008), pp. 3–27
2. R. Bowerman, B. Hall, P. Calamai, A multi-objective optimization approach to urban school bus routing: Formulation and solution method. *Transp. Res. A* **29**, 123–197 (1995)
3. A. Corberan, E. Fernandez, M. Laguna, R. Marti, Heuristic solutions to the problem of routing school buses with multiple objectives. *J. Oper. Res. Soc.* **53** 427–435 (2002)
4. T.G. Crainic, S. Mancini, G. Perboli, R. Tadei, A GRASP with path-relinking for the two-echelon vehicle routing problem, in *Advances in Metaheuristics*, ed. by L. Di Gaspero, A. Schaerf, T. Stutzle (Springer, Berlin, 2013), pp. 113–125
5. T. Feo, M. Resende, Greedy randomized adaptive search procedures. *J. Glob. Optim.* **6**, 109–133 (1995)
6. P. Festa, M. Resende, An annotated bibliography of GRASP - part I: algorithms. *Int. Trans. Oper. Res.* **16**, 124 (2009)
7. P. Festa, M. Resende, An annotated bibliography of GRASP - part II: applications. *Int. Trans. Oper. Res.* **16**, 131–172 (2009)
8. M. Gendreau, J. Potvin, *Handbook of Metaheuristics*, 2nd edn. (Springer, New York, 2010)
9. I. Giannikos, A multiobjective goal programming model for locating treatment sites and routing hazardous wastes. *Eur. J. Oper. Res.* **104**, 333–342 (1998)
10. C. Groer, B.L. Golden, E. Wasil, The consistent vehicle routing problem. *Manuf. Serv. Oper. Manage.* **11**(4), 630–643 (2009)
11. N. Jozefowicz, F. Semet, T. El-Ghazali, Multi-objective vehicle routing problems. *Eur. J. Oper. Res.* **189**, 293–309 (2008)
12. P. Lacomme, C. Prins, M. Sevaux, A genetic algorithm for a bi-objective capacitated arc routing problem. *Comput. Oper. Res.* **33**, 3473–3493 (2006)
13. T.-R. Lee, J.H. Ueng, A study of vehicle routing problem with load balancing. *Int. J. Phys. Distrib. Logist. Manag.* **29**, 646–648 (1999)

14. S. Mancini, Optimizing real-life freight-distribution problems. *Supply Chain Forum Int. J.* **15**(4), 42–50 (2014)
15. S. Mancini, Time dependent travel speed vehicle routing and scheduling on a real road network: the case of Torino. *Transp. Res. Proc.* **3**, 433–441 (2014)
16. W. Sessomboon, K. Watanabe, T. Irohara, K. Yoshimoto, A study on multi-objective vehicle routing problem considering customer satisfaction with due-time (the creation of Pareto optimal solutions by hybrid genetic algorithm). *Trans. Jpn. Soc. Mech. Eng.* (1998)
17. P. Toth, D. Vigo, *The Vehicle Routing Problem* (Society for Industrial and Applied Mathematics, Philadelphia, 2002)
18. P. Toth, D. Vigo, The granular tabu search and its application to the vehicle-routing problem. *INFORMS J. Comput.* **15**(4), 333–346 (2003)
19. K.G. Zografos, K.N. Androustopoulos, A heuristic algorithm for solving hazardous material distribution problems. *Eur. J. Oper. Res.* **152**, 507–519 (2004)

Chapter 18

Strip Algorithms as an Efficient Way to Initialise Population-Based Metaheuristics

Birsen İrem Selamoğlu, Abdellah Salhi, and Muhammad Sulaiman

Abstract The Strip Algorithm (SA) is a constructive heuristic which has been tried on the Euclidean Travelling Salesman Problem (TSP) and other planar network problems with some success. Its attraction is its efficiency. In its simplest form, it can find tours of length $\Omega(\sqrt{n})$ in $O(n \log n)$ operations where n is the number of nodes. Here, we set out to investigate new variants such as the 2-Part Strip Algorithm (2-PSA), the Spiral Strip Algorithm (SSA) and the Adaptive Strip Algorithm (ASA). The latter is particularly suited for Euclidean TSPs with non-uniform distribution of cities across the grid; i.e. problems with clustered cities. These cases present an overall low density, but high localised densities. ASA takes this into account in that smaller strips are generated where the density is high. All three algorithms are analysed, implemented and computationally tested against each other and the Classical Strip Algorithm. Computational results are included.

Keywords Strip Algorithm • Travelling Salesman • Heuristics • Optimisation

18.1 Introduction

Cheap but not necessarily robust algorithms are always needed. Here, the need is for such an algorithm to provide us with good approximate solutions to start with better performing population-based algorithms for intractable problems such as the TSP.

Often, hard problems encountered in the real world are not required to be solved exactly. Approximate solutions are enough particularly when execution time is crucial. Real world problems often require no more than a cheap but quick and reliable

B.İ. Selamoğlu (✉) • A. Salhi
University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK
e-mail: bisela@essex.ac.uk; as@essex.ac.uk

M. Sulaiman
Department of Mathematics, Abdul Wali Khan University, Mardan, Khyber Pakhtunkhwa, Pakistan
e-mail: msulaiman@awkum.edu.pk

algorithm that can deliver good approximate solutions. The TSP is one of those problems that occurs frequently in the real world. So, cheap algorithms were devised for it. A class of such algorithms is known as the Strip Algorithms (SA) [4, 2, 23]. They are simple and efficient heuristics for large Euclidean TSPs [14]. Their approach is to cut the plane into vertical or horizontal strips, then group the cities within strips according to their coordinates, and find Euclidean distances by following the order of the cities. Eventually the total length of a tour is obtained. The Strip Algorithm can be implemented in various ways. Furthermore, we believe that SA has a lot of scope for improvement.

This paper is organised as follows. In Sect. 18.2 ideas behind the SA are given. In Sect. 18.3, the 2-Part Strip Algorithm is introduced. In Sect. 18.4, the worst case performance of the new heuristic is analysed. In Sect. 18.5, two more variants of the strip algorithm are introduced briefly. Finally, results of experiments and the conclusion are given in Sect. 18.6.

18.2 Ideas Behind the Strip Algorithm

Consider the Euclidean TSP, which is a TSP, [16], cast over the Euclidean 2-dimensional space [11]. The idea is to connect nodes which are within a reduced area of the $2d$ problem. SA is not very different from approaches which build tours from minimum spanning trees and other greedy methods [9]. However, it allows for “less obvious” links to be taken into consideration and discourages criss-crossings from appearing in the tour.

Besides well-known construction heuristics such as, the nearest neighbour, the greedy algorithm, and insertion heuristics, there are different approaches for solving large TSP instances. In some situations, i.e. for large TSP instances, the speed of an algorithm becomes the priority rather than the quality of solutions. In other words, time is critical for some applications [12]. Therefore, for large Euclidean TSP instances, and for TSP instances in general, [18, 16], various algorithms have been introduced. Some of these heuristics can be listed as Space Filling Curves, Strip Heuristics and Decomposition Approaches [14, 10].

Few [4] has proved that for n points, with $n \geq 2$, the path length through them cannot be larger than $\sqrt{2n} + 1.75$ by using the strip idea. Beardwood et al. [2] have studied the shortest path for large numbers of points on a k -dimensional Euclidean space. They have assumed $k = 2$. They have shown that the optimum tour length is almost always proportional to \sqrt{nv} , where n is the number of points and v is the area of the plane defined. So, in the unit square this proportion has been defined as

$$\lim_{n \rightarrow \infty} \frac{c_{\text{opt}}}{\sqrt{nv}} = C, \quad (18.1)$$

where c_{opt} is the optimal solution, and $C \leq 0.9212$ is a constant.

In [19], the constant C found by Beardwood et al. was improved and the new constant was defined as $C \leq 0.894$ again by using the strip idea. Supowit et al. [22, 23] have studied the length of the shortest TSP tour in the worst case, through

n cities in the unit square. They have found the worst case tour length to be $\alpha\sqrt{n} + o(\sqrt{n})$, where $1.075 \leq \alpha \leq 1.414$. They introduced the strip algorithm which was inspired from the study in [2]. In their strip heuristic, the unit square is divided into $r = \lceil (\sqrt{\frac{n}{2}}) \rceil$ vertical strips. The configuration of their strip algorithm can be seen in Fig. 18.1. The first tour T_1 is formed by linking the points starting from the city which has the smallest y-axis value on the leftmost strip. The path goes up along that strip, then jumps to the next strip and goes down, and so on. The last city is connected to the first one to complete a tour. The second tour T_2 is formed in the same way. However, in order to construct the second tour the strips are shifted by $\frac{1}{2r}$ while the width of each strip is kept the same. The algorithm returns the tour with shortest length [23]. The complexity of this algorithm is $O(n \log n)$ [1].

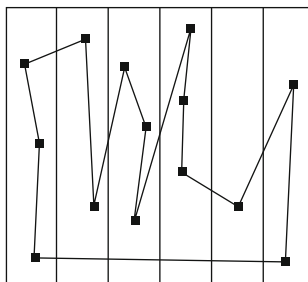


Fig. 18.1 An illustration of the classical strip algorithm

In [3], Dazango has introduced a new strategy for the expected length of tours by considering zones of various shapes and the density of vertices. The expected tour length was given as

$$D \approx \phi \sqrt{AN} \quad (18.2)$$

as $N \rightarrow \infty$, where N is the number of points uniformly distributed in the plane, A is the area of the plane, and ϕ is accepted to be 0.75 for the Euclidean metric.

In [8], Karloff has introduced the Local Strips method and proved that the upper bound for the shortest length of a TSP tour for n cities in a unit square cannot be larger than $\alpha\sqrt{n} + 11$, where n is the number of points and $\frac{\alpha}{\sqrt{(2)}} < 0.984$. Reinelt et al. [14], defined the strip algorithm as follows.

1. Split the planar graph into s vertical strips of equal width.
2. Split the planar graph into s vertical strips each having the same number of cities.
3. Split the planar graph into s horizontal strips of equal height.
4. Split the planar graph into s horizontal strips each having the same number of cities.
5. Choose the one which gives the best result.

The running time for this algorithm is estimated to be $\Theta(n \log n)$, and the number of strips is $s = \frac{\sqrt{n}}{2}$, where n indicates the total number of points. In [7], Johnson et al. modified the strip algorithm to overcome the problem that occurs

with clustered datasets. They ran the algorithm twice for each problem; once using vertical strips and once horizontal strips. They, then, chose the one which gives the better result. This method is called “2-Way Strip Algorithm”.

18.2.1 The Appropriate Number of Strips

In their algorithm, Supowit et al. [23], used $r = \lceil \sqrt{\frac{n}{2}} \rceil$ strips for the vertices uniformly distributed on a unit square. We applied this algorithm to some TSP problems taken from [13], but we used different numbers of strips chosen arbitrarily. We also used the number of strips suggested in the original paper for comparison purposes. Results can be seen in Table 18.1. Instances with 100, 200, 280, 442, 575 and 1084 points have been solved. The numbers of strips were chosen arbitrarily as 12, 20, 30, 40. But, we also used $\lceil \sqrt{\frac{n}{2}} \rceil$ strips. Average error is calculated as follows

$$Avg. Error = \frac{Avg. Solution Found - Optimum Solution}{Optimum Solution} * 100 \tag{18.3}$$

Table 18.1 The classical strip algorithm with various numbers of strips^a

Number of strips		r=12	r=20	r=30	r=40	r= $\lceil \sqrt{(n/2)} \rceil$
TSP instances	Opti.					
rd100	7910	63.80	129.45	188.26	231.51	33.93
kroA200	29,368	37.67	60.55	92.90	133.30	38.70
a280	2579	57.31	51.49	84.18	119.83	57.31
pcb442	50,778	37.49	56.22	82.83	126.42	48.77
rat575	6773	60.54	38.95	39.72	57.63	38.29
vm1084	239,297	73.83	53.23	55.65	77.23	49.19

^aAverage errors for different r values are given in %

Results show that, better approximations can be obtained using different numbers of strips. For kroA200 and pcb442 the algorithm with 12 strips gives better results. For 280 points, 20 strips give a better result. Reinelt et al. [14] defined the number of strips as $r = \frac{\sqrt{n}}{2}$, where n indicates the total number of nodes. Another study of the optimum number of strips can be found in [3]. Dazango claimed that, for any rectangle containing the cities, the width of strips, i.e. their number, affects the quality of solution. If width w is too small then there will be extra length to connect points. But, if w is too large, then there will be zigzags which increase the tour length. For the configuration in his study, where A is the area and σ is the number of points in unit area, the optimum width was found to be $w = \sqrt{\frac{3A}{\sigma}}$. Dazango’s configuration of the strip algorithm can be seen in Fig. 18.2.

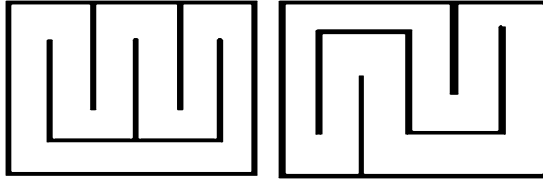


Fig. 18.2 Dazango's configuration of the strip algorithm

18.3 2-PSA: The 2-Part Strip Algorithm

The difference between CSA, [23], and 2-PSA is that in the latter the plane is divided into two by one horizontal line. Furthermore, the number of strips can be changed in the given interval, so more alternatives can be compared in one method. In practical implementations, this algorithm aims at minimising the distance between the last visited point and the starting point. In both upper and lower part CSA is applied. It starts from the bottom part and follows the ascending or descending order of y -axis values to connect to the next strip. When it goes through the upper part, the same procedure is used and the last point meets the starting point. The time complexity of this algorithm is $O(n \log n)$, because of the sorting that has been performed with respect to the vertical coordinates. The 2-PSA can be described as follows:

1. Divide the grid into two horizontal parts each having roughly equal number of points; then divide each into r vertical strips.
2. Proceed as in the basic strip algorithm from either of the horizontal parts, but reverse the sense of travel when at the end of the first part.
3. Connect the last point of the tour to the starting point to complete the Hamiltonian tour.
4. Return the tour and its length.

Note that r can be chosen between $r = \lceil \sqrt{(n/2)} \rceil$ and width $w = \sqrt{\frac{3A}{\sigma}}$ or after some experimentation.

18.4 Worst-Case Analysis of 2-PSA and an Upper Bound on the Minimum Tour Lengths Returned

In this section, 2-PSA is analysed. Since it is a refined version of CSA, [23], the analysis of 2-PSA has been carried out in the same way. The L_2 metric has been used. An example output of 2-PSA for a large TSP instance can be seen in Fig. 18.3.

Let there be n points uniformly distributed in the unit square. Construct two tours, T_1 and T_2 according to the procedure described in Sect. 18.3 and let the lengths of these tours be LT_1 and LT_2 , respectively. In order to define upper bounds on the optimum tour length, create two tours BT_1 and BT_2 , the former following the median

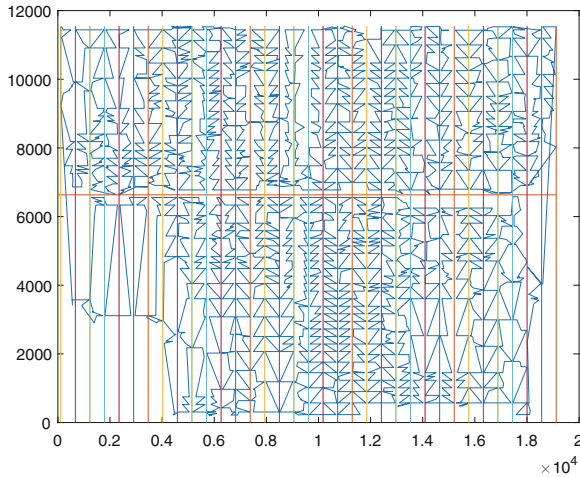


Fig. 18.3 The graphical representation of the solution found by 2-PSA for TSP problem r15915 [13]

of a strip that is used to construct T_1 and the latter that of a shifted strip to the right by $\frac{1}{2r}$, that is used to construct T_2 . Tours BT_1 and BT_2 are generated using the same procedure as that of 2-PSA. However, since each path goes up or down through the median of the strip and its shifted counterpart, in order to connect the points, there is a jut that goes horizontally to the middle of the intersection of the two strips. This can be seen in Fig. 18.4. A similar representation can be found in [23].

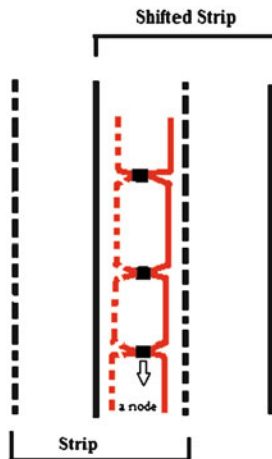


Fig. 18.4 The paths BT_1 and BT_2 visiting points

Assuming that the horizontal line cuts the unit square from the midpoint of the leftmost and the rightmost sides, the vertical length of the upper and lower parts becomes $\frac{1}{2}$ each. The total length of BT_1 and BT_2 are calculated in order to define an upper bound for the 2-PSA. Since the total vertical length for BT_1 is $\frac{r}{2}$ for the lower part and $\frac{r}{2}$ for the upper part, it is therefore, r for the total tour length. Similarly for BT_2 , it is $\frac{r+1}{2} + \frac{r+1}{2} = r + 1$. Since the strips have been shifted to construct the second tour, the total number of strips has increased by 1. The total horizontal length for BT_1 to connect one strip to another is, $1 - \frac{1}{r}$ for the lower part, because on the leftmost and the rightmost strips, there are gaps of $\frac{1}{2r}$, and the same applies for the upper part, therefore the total tour length would be $2 - \frac{2}{r}$. For BT_2 , this length is 2. In order to calculate the total horizontal length of visiting each point and coming back to the median line, let us assume that points have been placed $\frac{1}{4r}$ apart from each half strip. Therefore, from each path, which is on the median, there is a horizontal length of $2\frac{1}{4r}$. For both BT_1 and BT_2 , the total horizontal length is $\frac{n}{r}$. Finally, the length of connecting the last visited point to the starting point is 1 in the worst case for both tours. By using the triangle inequality and assuming $r = \lceil (\sqrt{\frac{n}{3}}) \rceil$, we can write

$$\begin{aligned}
 LT_1 + LT_2 &\leq \text{length}(BT_1) + \text{length}(BT_2), \\
 &\leq r + r + 1 + \frac{n}{r} + (2 - \frac{2}{r}) + 2 + 1 + 1, \\
 &\leq \frac{n}{r} + 2r + O(1), \\
 &\leq 5\frac{\sqrt{3n}}{3} + O(1).
 \end{aligned} \tag{18.4}$$

Therefore, as in [23], either LT_1 or LT_2 is less or equal to $5\frac{\sqrt{3n}}{6} + O(1)$ which is the worse upper bound in terms of cost. However, experiments show that assuming that we know the optimal width of the strips, we can deduce that

$$\text{length}(T_1^{2-PSA}) \leq \text{length}(T_1^{CSA}).$$

In other words, 2-PSA generates better tours than CSA. But, referring to experimental evidence again, CSA is much faster (see Table 18.4).

18.5 Other Implementations of the Strip Algorithm

In this study, different variants of the strip algorithm have been developed and tested. However, amongst all variants, only 2-PSA produced competitive results. In this section, we consider these variants.

18.5.1 *The Adaptive Strip Algorithm (ASA)*

The classical strip algorithm is not effective on instances with clustered cities. To generate good quality initial tours for such TSP instances in a cheap way, a new idea has been explored. This idea consists in generating strips with approximately similar numbers of cities in them. A threshold of cities is arbitrarily chosen and then any strip with a number of cities greater than this threshold is subdivided into two new strips. The process is started with the initial grid containing all cities. Since the threshold is set well below the number of given cities, this initial grid or box is vertically split into two. The choice of splitting vertically first is arbitrary. Each resulting strip or box is then checked for the number of cities it has. If this number is higher than the threshold, the box is split further. The process continues. Note that any box which is a result of a vertical split is split horizontally, and any box which is the result of a horizontal split is divided vertically. This approach results in small boxes where there is a high density of cities and larger ones where the density is low. In each box, a representative city is then chosen at random. These representatives are linked up using CSA. Then in every box a path linking all cities is generated. These paths fall into the overall path which links the boxes, to form a Hamiltonian path. The last city in the last box is then linked to the first city in the first box to complete a tour. Note that, boxes with no cities in them are ignored. The Adaptive Strip Algorithm of ASA can be described as follows.

1. Split the unit square into horizontal and vertical strips. Each strip should contain a predetermined number of cities at most.
2. Choose a city from each strip as a representative of the cities in that strip.
3. Apply CSA to all representative cities to link them into a path,
4. Apply CSA to all cities in each strip.
5. Link up all paths within strips to each other to create the final tour.

The graphical result of ASA for solving greece9882 problem can be seen in Fig. 18.5. A 50-city TSP problem was solved by using both ASA and CSA. Graphical results can be seen in Fig. 18.6a, b. In this set of cities, there are clusters, which cause CSA to work ineffectively. Applying ASA, the same problem was solved by hand. The total length of the tour formed with CSA is 6.33. With ASA it is 5.50. This shows that the clustered TSP instances can be solved more efficiently using the new algorithm.

18.5.2 *The Spiral Strip Algorithm (SSA)*

In this approach, the plane is cut into a number of horizontal and vertical strips proportional to the number of cities. Assume that r is the number of vertical strips and p the number of horizontal strips. Therefore, $r * p$ cells are created. Both r and p are calculated as $\lceil \frac{\text{number of cities}}{t} \rceil$, where t is a positive integer used as the number of strips. The application starts from the leftmost upper corner cell and follows a spiral

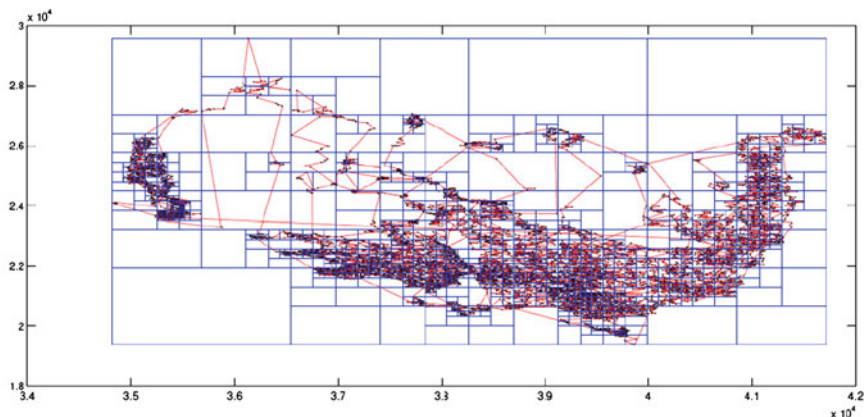


Fig. 18.5 ASA solution of greece9882

way and ends up about the middle of the plane. The Hamiltonian cycle is completed by connecting the ending point with the starting one.

One similar approach can be found in [6]. To justify the investigation of the SSA, we compare the tour length it produces with that of the CSA, for instance. Let consider two identical unit squares with n number of cities spread over each. In order to complete the Hamiltonian cycle, let solve each using CSA and SSA, respectively. If we use r strips to solve the problem with CSA, the total tour length in the worst case will be $r + \frac{r-1}{r} + \sqrt{2}$. Similarly, to solve the problem using SSA with r vertical and r horizontal strips, then, the total tour length in the worst case will be $\frac{3}{r}(r-1) + \frac{(r-2)(r-1)}{r} + \frac{\sqrt{2}}{2}$. In both cases, it is assumed that nodes are exactly on the strips. Comparison of the results of each problem proves that

$$\text{length}(T^{SSA}) \leq \text{length}(T^{CSA}). \quad (18.5)$$

An example of the implementation can be seen in Fig. 18.7.

18.6 Computational Results and Conclusion

We have implemented 2-PSA and other similar schemes and tested them on standard TSP problems ranging from 51 to 5915 cities [13]. The computing platform is an Intel core I5 PC with a 3.40 GHz processor and 16 Gigabytes RAM, running Windows 7. All algorithms are coded in Matlab R2014a. The results show the value of the strip approach at least as a tool for generating computationally cheap but better tours than those generated randomly. Note that it was not expected to generate solutions too close to the optimum, hence the relatively large errors observed. The aim is to have something which is better than just random tours. Speed is the essence. For comparison purposes, the CPU time of 2-PSA for the TSP with 9882

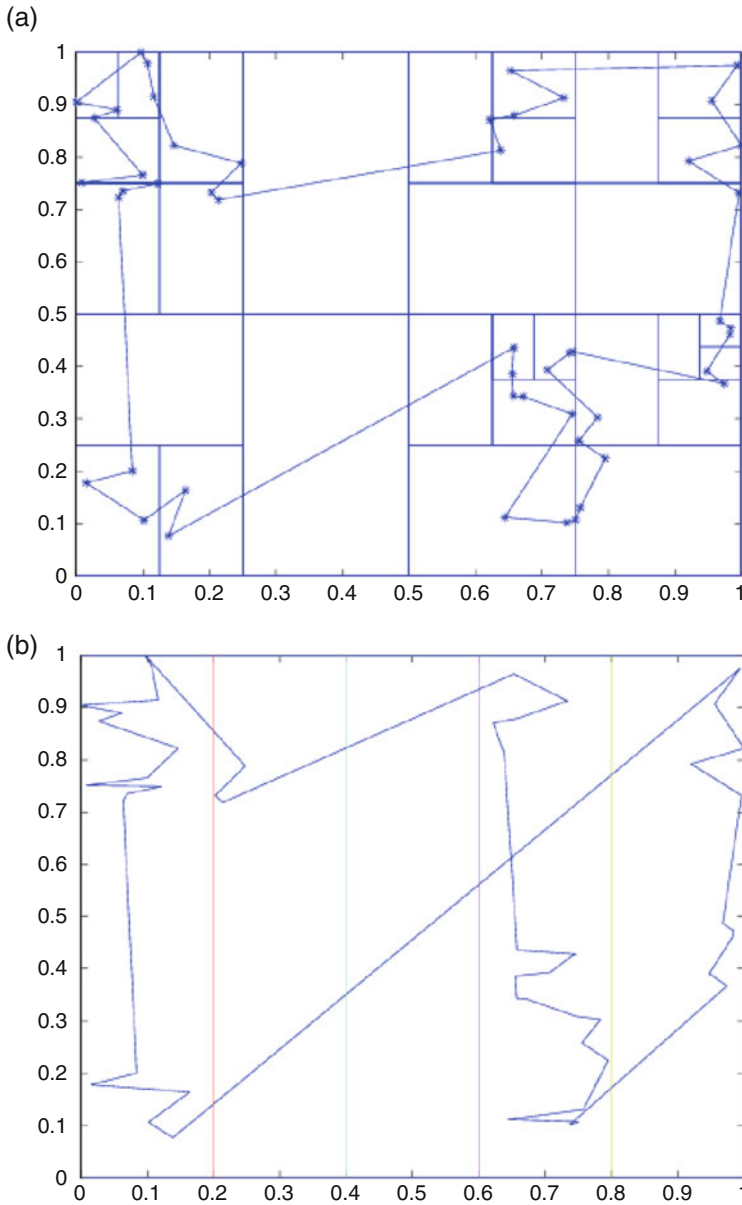


Fig. 18.6 (a) The adaptive strip algorithm on the 50-city problem. (b) The classical strip algorithm on the 50-city problem

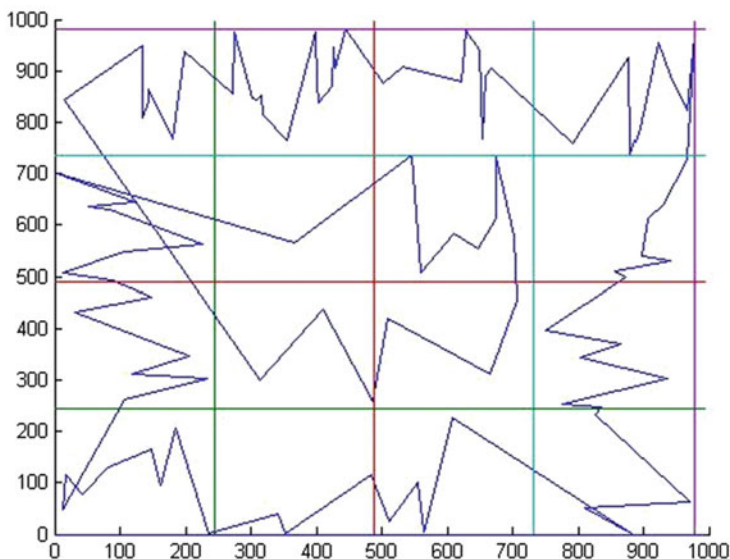


Fig. 18.7 SSA solution of problem rd100 [13]

cities is less than $\frac{1}{10}$ th of a second, while that of the greedy algorithm is around 8 min. Comparative results between the different variants of the strip algorithm and random permutation considered have also been recorded in Table 18.2; the superiority of 2-PSA over CSA, SSA and the random permutation is very clear. Note that the paper has also introduced new strip algorithms and analysed them in the worst case. Further work will concentrate hybrids which may be slower but that can produce better tours than at the moment. A hybridisation of ASA and a greedy algorithm seems to produce much better tours. Accounts of this current work will be reported in the future.

Table 18.2 Deviation (in %) from the optimum of CSA, SSA, 2-PSA and random permutation

TSP instances	No. of cities	Opt. sol.	CSA	SSA	2-PSA	Rand. perm.
eil51	51	426	21.67	40.63	19.62	303.28
st70	70	675	31.94	47.93	17.75	443.46
rd100	100	7910	32.84	48.18	23.80	598.31
a280	280	2579	41.50	62.19	29.71	1190.02
rat575	575	6773	47.70	37.54	20.83	1604.73
vm1084	1084	239,297	51.74	519.27	40.38	3498.61
vm1748	1748	336,556	55.58	671.69	44.19	4342.15
rl5915	5915	565,530	78.24	163.03	64.11	7436.97

In Table 18.3, real-life instances [24] have been used to compare the performance of CSA, 2-PSA, ASA and the random permutation. Note that 2-PSA and

ASA give better results than CSA. Table 18.4 records the total computation time for each algorithm. ASA has solved all problems in a reasonable time. Results show that 2-PSA runs 7–20 times longer than CSA, however it reduces the error from 10% to mostly half. ASA is faster than 2-PSA, and for large instances, it gives better tour lengths than both CSA and 2-PSA. Results demonstrate that although the random permutation is the fastest, it gives the worst approximation amongst the others.

Table 18.3 Deviation (in %) from the optimum of CSA, 2-PSA, ASA and random permutation

TSP instances	No. of cities	Opt. sol.	CSA	2-PSA	ASA	Rand. perm.
usca50	50	14,497	108.79	54.69	97.61	451.08
zimbabwe929	929	95,345	68.13	57.27	61.91	1319.46
canada4663	4663	1,290,319	158.17	114.88	86.11	1793.95
greece9882	9882	300,899	90.81	90.12	87.75	12,529.2

Table 18.4 CPU time of each algorithm

CPU time (s)	usca50	zimb929	ca4663	gre9882
CSA	0.008	0.003	0.006	0.012
2-PSA	0.028	0.061	0.117	0.083
ASA	0.006	0.009	0.020	0.047
Random permutation	0	0	0	0.001

The Strip Heuristic is cheap yet effective in finding good tours in a short time. The proposed algorithms, 2-PSA and ASA have given better results than CSA. Although the returned solutions are far from the optimum solutions in terms of quality, they have been obtained quickly. This makes them potential providers of initial populations for other meta-heuristics such as the Plant Propagation Algorithm or the Strawberry Algorithm [17, 21, 20], the Genetic Algorithm [5, 15], and others.

Acknowledgement We are grateful to the Ministry of National Education of the Republic of Turkey for sponsoring this work.

References

1. D. Avis, A survey of heuristics for the weighted matching problem. *Networks* **13**(4), 475–493 (1983)
2. J. Beardwood, J.H. Halton, J.M. Hammersley, The shortest path through many points, in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 55 (Cambridge University Press, Cambridge, 1959), pp. 299–327
3. C.F. Daganzo, The length of tours in zones of different shapes. *Transp. Res. B Methodol.* **18**(2), 135–145 (1984)
4. L. Few, The shortest path and the shortest road through n points. *Mathematika* **2**(02), 141–144 (1955)

5. J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence* (MIT Press, Cambridge, 1992)
6. M. Iri, K. Murota, S. Matsui, Heuristics for planar minimum-weight perfect matchings. *Networks* **13**(1), 67–92 (1983)
7. D.S. Johnson, L.A. McGeoch, Experimental analysis of heuristics for the STSP, in *The Traveling Salesman Problem and Its Variations* (Springer, Heidelberg, 2007), pp. 369–443
8. H.J. Karloff, How long can a Euclidean traveling salesman tour be? *SIAM J. Discret. Math.* **2**(1), 91–99 (1989)
9. G. Laporte, The traveling salesman problem: an overview of exact and approximate algorithms. *Eur. J. Oper. Res.* **59**(2), 231–247 (1992)
10. G. Laporte, A concise guide to the traveling salesman problem. *J. Oper. Res. Soc.* **61**(1), 35–40 (2010)
11. S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.* **21**(2), 498–516 (1973). DOI 10.1287/opre.21.2.498. <http://dx.doi.org/10.1287/opre.21.2.498>
12. Y. Marinakis, Heuristic and metaheuristic algorithms for the traveling salesman problem, in *Encyclopedia of Optimization* (Springer, Boston, 2009), pp. 1498–1506
13. G. Reinelt, TSPLIB - A T.S.P. library. Technical Report 250, Universität Augsburg, Institut für Mathematik, Augsburg (1990)
14. G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications* (Springer, Berlin, 1994)
15. J.A.V. Rodríguez, A. Salhi, Hybrid evolutionary methods for the solution of complex scheduling problems, in *Advances in Artificial Intelligence Research in Computing Science*, vol. 20, eds. by A. Gelbukh, S. Torres, I. López (Center for Computing Research of IPN, 2006), pp. 17–28
16. A. Salhi, The ultimate solution approach to intractable problems, in *Proceedings of the 6th IMT-GT Conference on Mathematics, Statistics and its Applications* (2010), pp. 84–93
17. A. Salhi, E. Fraga, Nature-inspired optimisation approaches and the new plant propagation algorithm, in *Proceedings of the ICeMATH2011* (2011), pp. K2-1–K2-8
18. A. Salhi, Ö. Töreayen, A game theory-based multi-agent system for expensive optimisation problems, in *Computational Intelligence in Optimization* (Springer, Berlin, 2010), pp. 211–232
19. S. Steinerberger, A new upper bound for the traveling salesman constant. *Adv. Appl. Probab.* **47**(01), 27–36 (2015)
20. M. Sulaiman, A. Salhi, E.S. Fraga, The plant propagation algorithm: modifications and implementation. *ArXiv e-prints* (2014)
21. M. Sulaiman, A. Salhi, B.I. Selamoglu, O.B. Kirikchi, A plant propagation algorithm for constrained engineering optimisation problems. *Math. Probl. Eng.* (2014). Article ID 627416, 10 pp. doi:10.1155/2014/627416
22. K.J. Supowit, D.A. Plaisted, E.M. Reingold, Heuristics for weighted perfect matching, in *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing* (ACM, New York, 1980), pp. 398–419
23. K.J. Supowit, E.M. Reingold, D.A. Plaisted, The travelling salesman problem and minimum matching in the unit square. *SIAM J. Comput.* **12**(1), 144–156 (1983)
24. Tsp test data. <http://www.math.uwaterloo.ca/tsp/data/>

Chapter 19

Metaheuristics for the Temporal Bin Packing Problem

Fabio Furini and Xueying Shen

Abstract We study an extension of the Bin Packing Problem, where items consume the bin capacity during a time window only. The problem asks for finding the minimum number of bins to pack all the items respecting the bin capacity at any instant of time. Both a polynomial-size formulation and an extensive formulation are studied. Moreover, various heuristic algorithms are developed and compared, including greedy heuristics and a column generation based heuristic. We perform extensive computational experiments on benchmark instances to evaluate the quality of the computed solutions with respect to strong bounds based on the linear programming relaxation of the proposed formulations.

Keywords Temporal bin packing problem • Heuristic algorithms • Column generation

19.1 Introduction

The Bin Packing Problem (BPP) [11] is a classical NP-hard combinatorial optimization problem and it has been widely studied in the literature, we refer the interested readers to [6, 5]. It has applications in different areas, such as filling up containers, placing data on multiple disks and many others. Numerous BPP variants have been studied as the Multi-Dimensional Bin Packing Problem or the On-line Bin Packing Problem. In this paper, we study a natural generalization of the BPP called *Temporal Bin Packing Problem (TBPP)*. The input of the problem are a bin capacity W and a set of items $N = \{1, 2, \dots, n\}$, where each item $i \in N$ is associated to an integer weight $w_i \leq W$ consuming the bin capacity during the time window $[s_i, t_i)$. The TBPP asks for finding the minimum number of bins to pack

F. Furini (✉)

PSL, Université Paris-Dauphine, 75775 Paris Cedex 16, France, CNRS, LAMSADE UMR 7243
e-mail: fabio.furini@dauphine.fr

X. Shen

PSL, Université Paris-Dauphine, 75775 Paris Cedex 16, France, CNRS, LAMSADE UMR 7243

Decision Brain, 18 rue Yves Toudic, 75010 Paris, France

e-mail: sylvia.shen@decisionbrain.com

all the items so that the capacity of each bin is not exceeded at any point in time. Despite the capacity requirements being defined on the entire time horizon, clearly it is sufficient to satisfy the capacity restrictions only at the starting time of each item. Let us define $S_j := \{i \in N : s_i \leq s_j \text{ and } t_i > s_j\}$ as the set of active items at time s_j , then for each bin, the total capacity of packed items that belong to S_j should be less than or equal to the bin capacity W for any $j \in N$. Moreover, if $S_j \subseteq S_k$, then the associated capacity constraint at time s_j is dominated by that of time s_k . Define $T = \{t \in N : S_t \not\subseteq S_k, \forall k \in N\}$, which represents the index set of all the non-dominated constraints, then we only need to consider the capacity usage at each $t \in T$, and we call t a *time step*.

In Fig. 19.1, we give one example to illustrate a TBPP instance with a bin capacity $W = 4$ and 5 items (weights $w_1 = 2, w_2 = 2, w_3 = 3, w_4 = 2$ and $w_5 = 1$). The starting time and ending time (s_i and t_i) are also shown in the Fig. 19.1. For instance, item 1 is active from time point 1 to 3 and item 2 is active from time point 2 to 14. The set of non-dominated time steps are $t = 2, t = 7$ and $t = 12$, with the following simultaneously active item sets $S_2 = \{1, 2\}, S_4 = \{2, 3, 4\}$ and $S_5 = \{2, 5\}$.

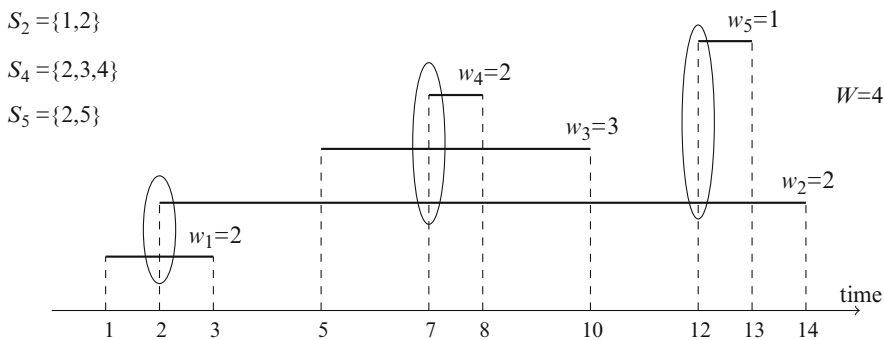


Fig. 19.1 Example of a temporal bin packing problem instance

In fact, the TBPP can be seen as a special case of the multi-dimensional bin packing problem where each time step corresponds to one dimension. Moreover, for each item $i \in N$, it either consumes zero capacity or w_i for each dimension. The concept of “temporal extension” has already been used to extend the classical knapsack problem, which is called Temporal Knapsack Problem (TKP) [1]. TKP has been shown to be strongly NP-hard in [2] and different exact algorithms have been developed, we address the interested reader to [1, 3].

19.1.1 Greedy-Type Algorithms

Inherited from the BPP, we extend two greedy-type algorithms for the TBPP: the first-fit algorithm and the best-fit algorithm.

- First Fit-Algorithm: pack the current item into the first non-empty bin in which it fits. If no such bin exists, pack the current item into an empty bin.
- Best Fit-Algorithm: pack the current item into the non-empty bin with the largest free space in which it fits. By “the largest free space”, we mean the summation of the remaining capacities at all the time steps. If no such bin exists, pack the current item into an empty bin.

These two algorithms are computationally efficient, therefore they can be used to provide the number of bins m for the Mixed Integer Programming (MIP) formulation (see the following sections), or to provide an initial solution for more advanced heuristic algorithms.

The rest of the paper is organized as follows. The compact formulation is given in Sect. 19.2, while an extensive formulation and a heuristic algorithm based on it are presented in Sect. 19.3. The computational results are given in Sect. 19.4, which is followed by the conclusion in Sect. 19.5.

19.2 Compact Formulation

Let $m \leq n$ be an upper bound on the number of bins necessary to fit all items. For each item $i \in N$ and bin $b \in M = \{1, 2, \dots, m\}$, we introduce two sets of binary variables:

$$x_i^b = \begin{cases} 1 & \text{if item } i \text{ is packed in bin } b, \\ 0 & \text{otherwise,} \end{cases} \quad y^b = \begin{cases} 1 & \text{if bin } b \text{ is used,} \\ 0 & \text{otherwise.} \end{cases}$$

Then the polynomial-size MIP formulation reads as follows:

$$Form_A \quad \min \sum_{b \in M} y^b \tag{19.1}$$

$$\sum_{b \in M} x_i^b \geq 1 \quad i \in N, \tag{19.2}$$

$$\sum_{i \in S_t} w_i x_i^b \leq W y^b \quad b \in M, t \in T, \tag{19.3}$$

$$x_i^b \in \{0, 1\} \quad i \in N, b \in M, \tag{19.4}$$

$$y^b \in \{0, 1\} \quad b \in M. \tag{19.5}$$

Exploiting the link between the TBPP and the BPP, we can compute a valid lower bound for the TBPP from the solution values of $|T|$ BPPs, one for each time step separately. The result is shown in the following property:

Property 19.1. Given an instance of the TBPP, for $t \in T$ we define the $BPP(t)$ as follows:

$$\begin{aligned}
 BPP(t) \quad & \min \sum_{b \in M} y^b \\
 & \sum_{b \in M} x_i^b \geq 1 && i \in N, \\
 & \sum_{i \in S_t} w_i x_i^b \leq W y^b && b \in M, \\
 & x_i^b \in \{0, 1\} && i \in N, b \in M, \\
 & y^b \in \{0, 1\} && b \in M,
 \end{aligned}$$

and let $z^*(BPP(t))$ be the optimal objective function value of $BPP(t)$. Then,

$$\max_{t \in T} z^*(BPP(t))$$

is a valid lower bound of the TBPP.

Proof. For any $t \in T$, the $BPP(t)$ results from relaxing the capacity constraints (19.3) for each $b \in M$ on all time steps except the one at time step t . Therefore, it is a relaxation problem of the TBPP and provides a valid lower bound. The result follows. \square

The following example shows that the optimal solution of the TBPP may differ from the optimal solutions of the $BPP(t)$ for each $t \in T$. Given the bin capacity $W = 10$, and 5 items with weights: $w_1 = 9, w_2 = 2, w_3 = 4, w_4 = 6$ and $w_5 = 8$. We consider 2 time steps defined as $S_1 = \{1, 2, 3\}$ and $S_2 = \{2, 3, 4, 5\}$. Then we have $\max_{t=1,2} z^*(BPP(t)) = 2$ (i.e., 2 bins are necessary at any time step), but the optimal solution value of the TBPP is 3.

Besides the lower bound given above, the Linear Programming (LP) relaxation of *Form_A* also provides a valid lower bound. Moreover, the following property shows that the LP relaxation of above formulation *Form_A* can be solved in a closed form.

Property 19.2. Given an instance of the TBPP, the optimal objective function value of the LP relaxation of the formulation *Form_A* can be given in the following closed form:

$$\bar{z}_{lp} = \frac{\max_{t \in T} \sum_{i \in S_t} w_i}{W}, \tag{19.6}$$

with the solution

$$\begin{aligned} \bar{x}_i^b &= \frac{1}{m} \quad i \in N, b \in M, \\ \bar{y}^b &= \frac{\max_{t \in T} \sum_{i \in S_t} w_i}{W \cdot m} \quad b \in M. \end{aligned}$$

Proof. First, we show that the solution is feasible. For any $t \in T$, $b \in M$,

$$\sum_{i \in S_t} w_i \bar{x}_i^b = \sum_{i \in S_t} w_i \cdot \frac{1}{m} \leq \max_{t \in T} \sum_{i \in S_t} w_i \cdot \frac{1}{m} = \frac{\max_{t \in T} \sum_{i \in S_t} w_i}{W \cdot m} \cdot W = W \bar{y}^b.$$

For any $i \in N$,

$$\sum_{b \in M} \bar{x}_i^b = \sum_{b \in M} \frac{1}{m} = 1.$$

Therefore, the solution (\bar{x}_i^b, \bar{y}^b) is feasible, and the objective function value can be calculated directly as \bar{z}_{lp} .

Second, we show that the solution is optimal. For any $t \in T$, we have $BPP(t)$ is a relaxation of $Form_A$, the LP relaxation of $BPP(t)$ is a relaxation of the LP relaxation of $Form_A$. Let $z_{lp}(Form_A)$ and $z_{lp}(BPP(t))$ denote the optimal objective function value of the LP relaxation of $Form_A$ and $BPP(t)$ respectively. Then

$$z_{lp}(Form_A) \geq z_{lp}(BPP(t)) = \frac{\sum_{i \in S_t} w_i}{W}.$$

The second equality is proved in [13]. Take $t^* = \operatorname{argmax}\{t \in T : \max_{t \in T} \sum_{i \in S_t} w_i\}$, the results follows. \square

The formulation $Form_A$ is compact in the sense that it has a polynomial number of variables and constraints. An MIP solver can be directly used to solve it, or can be used to obtain heuristic solutions by imposing a given time limit. However, the performance of the formulation highly depends on the upper bound m due to its impact on the number of variables and constraints. Also, the symmetric structure of the model generates a huge amount of equivalent solutions. For large scale problems, MIP solvers are not able to find optimal solutions within reasonable amount of time. Due to these limitations, we propose an extensive formulation which, together with column generation techniques, can be used to develop exact algorithms and heuristic algorithms. Column generation is a powerful tool to deal with LP problems with exponential number of variables. It has been successfully applied to many classes of problems including the cutting stock problem [8, 9], the vehicle routing problem [4], the crew scheduling problem [12] and many others. In the following of the paper, we focus on an extensive formulation of the problem and a heuristic algorithm

based on column generation. We also computationally show that the LP relaxation of the extensive formulation provides tighter bounds, which can be used to prove the optimality of heuristic solutions.

19.3 Extensive Formulation

Let us introduce the extensive formulation with an exponential number of variables associated with all feasible packing patterns, i.e., subsets of items respecting bin capacities at any point in time. Let \mathcal{S} represent the collection of all possible feasible packing patterns:

$$\mathcal{S} = \left\{ S \subseteq N : \sum_{i \in S_t \cap S} w_i \leq W, \forall t \in T \right\}.$$

For each $S \in \mathcal{S}$, we introduce a binary variable z_S :

$$z_S = \begin{cases} 1 & \text{if pattern } S \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

Then the extensive formulation of the TBPP reads as follows:

$$Form_B(\mathcal{S}) \quad \min \sum_{S \in \mathcal{S}} z_S \quad (19.7)$$

$$\sum_{S \in \mathcal{S}: i \in S} z_S \geq 1 \quad i \in N, \quad (19.8)$$

$$z_S \in \{0, 1\} \quad S \in \mathcal{S}. \quad (19.9)$$

The LP relaxation of $Form_B(\mathcal{S})$ is formulated as ((19.7)-(19.8), (19.10)):

$$z_S \geq 0 \quad S \in \mathcal{S}, \quad (19.10)$$

and it is denoted by $MLP(\mathcal{S})$. The model needs column generation techniques to be efficiently managed. Actually, it has exponentially many variables, which cannot be explicitly generated for large-size instances.

To solve the LP relaxation $MLP(\mathcal{S})$, we start with a restricted problem $MLP(\mathcal{S}')$ with an initial feasible set of variables associated to $\mathcal{S}' \subseteq \mathcal{S}$. The dual problem of $MLP(\mathcal{S}')$ reads as follows:

$$\begin{aligned}
 DLP(\mathcal{S}') \quad & \max \sum_{i \in N} \pi_i \\
 & \sum_{i \in S} \pi_i \leq 1 \quad S \in \mathcal{S}', \\
 & \pi_i \geq 0 \quad i \in N.
 \end{aligned}$$

Finding a variable with negative reduced cost to be added to $MLP(\mathcal{S}')$ is equivalent to find a violated constraint in $DLP(\mathcal{S}')$. Given an optimal dual solution $\pi^* = \{\pi_i^*\}$ to $MLP(\mathcal{S}')$, the separation problem asks to determine a subset $S^* \subseteq \mathcal{S}$ such that $\sum_{i \in S^*} \pi_i^* > 1$. The problem arising during the separation procedure is a 0-1 TKP. This problem can be modelled using a binary variable x_i which takes value 1 if item $i \in N$ is selected in subset S^* and it reads as follows:

$$\max \sum_{i \in N} \pi_i^* x_i \tag{19.11}$$

$$\sum_{i \in \mathcal{S}_t} w_i x_i \leq W \quad t \in T, \tag{19.12}$$

$$x_i \in \{0, 1\} \quad i \in N. \tag{19.13}$$

If a feasible pattern S^* with maximum profit greater than 1 is found, the column corresponding to S^* is added to the current $MLP(\mathcal{S}')$. When no feasible pattern with maximum profit greater than 1 is found, $MLP(\mathcal{S})$ is optimally solved, and the rounded-up value of the optimal objective function value to $MLP(\mathcal{S}')$ is a valid lower bound for the TBPP.

In order to find an optimal integer solution of model $Form_B(\mathcal{S})$, a branch-and-price algorithm is necessary, i.e., the column generation procedure must be performed at each node of the branching scheme. The development of such an algorithm goes beyond the scope of this paper, we instead use this formulation to derive a heuristic algorithm and strong dual bounds for hard TBPP instances.

19.3.1 Column Generation Heuristics

In order to obtain good feasible integer solutions, we test a heuristic strategy based on formulation $Form_B(\mathcal{S})$. The procedure consists of applying an MIP solver to model $Form_B(\mathcal{S}')$, by considering all the variables generated to optimally solve its LP relaxation ($MLP(\mathcal{S}')$). This strategy takes full advantage of the advanced routines embedded in MIP solvers. The pseudo-code of our heuristic algorithm is given in Fig. 19.2.

begin

Initialize the set \mathcal{S}' with the initial heuristic solution.

1. Solve the $MLP(\mathcal{S}')$ by column generation, and update set \mathcal{S}' by adding the generated columns (cutting patterns);
2. Solve $Form_B(\mathcal{S}')$ with an MIP solver by considering the subset of variables $z_S, S \in \mathcal{S}'$.

end.

Fig. 19.2 Heuristic algorithm based on column generation

The idea of generating a subset of variables containing at least one feasible solution, and then solving the corresponding restricted model with an MIP solver, is an effective method to derive strong upper and lower bounds for the TBPP (see the next section of this manuscript for further details). Moreover, this framework can be applied to different models with an exponential number of variables. We address the interested reader to [7], where similar ideas have been applied to tackle 2-Dimensional BPP.

19.4 Computational Experiments

In this section, we present our computational results for the heuristic algorithms and lower bounds developed above. All the algorithms described in the previous sections are implemented in C++ and run on one core of an Intel Core i7-4790 2.50 GHz 3.60, with 16 GB shared memory, under the Linux Ubuntu 12.4 operating system. All the LP problems and the MIP problems are solved with CPLEX 12.6 [10]. In the following tables, all the computing times are expressed in seconds.

As testbed, we use the instances of the TKP presented in [3], due to the fact that the TBPP instances share the same data structure with those of the TKP, except the absence of item profits. The testbed consists of two types of instances, which are called the I instances and the U instances respectively. Each set consists of one hundred instances which are further divided into ten classes based on the different values of instance generating parameters (see [3] for further details).

We apply the two greedy-type heuristic algorithms as well as the $Form_A$ with a given time limit of 5 min. The detailed results for each instance can be found in Tables 19.1 and 19.2. For each instance, the number of items n and the number of time steps $|T|$ are reported. Moreover, the objective function value and computation time are given in columns *Obj* and *Time*. For comparison, we summarize the results in Table 19.3. In this table, for each class of instances, the average number of items and average number of time steps are reported to illustrate the problem size. We present the average objective function values to give an idea on the number of bins necessary to solve the instances. Clearly the model $Form_A$ benefits from low values of m . Also given are the average computational time and the average gap. The column *Exit Gap* reports the average ratio $\frac{UB-LB}{UB}$ where UB and LB are the primal and the dual bounds computed by CPLEX respectively at time limit (0.00 means instances

Table 19.1 Heuristic algorithms comparison and computational behavior of formulation *Form_A*: Instance U

Inst	n	T	<i>Form_A</i>		Best fit		First fit		Inst	n	T	<i>Form_A</i>		Best fit		First fit	
			Obj	Time	Obj	Time	Obj	Time				Obj	Time	Obj	Time	Obj	Time
U1	1000	459	2	0.02	2	0.01	2	0.00	U51	1000	445	6	0.20	6	0.00	6	0.01
U2	1000	468	2	0.02	2	0.01	2	0.00	U52	1000	467	5	0.18	6	0.01	5	0.00
U3	1000	485	2	0.02	2	0.00	2	0.00	U53	1000	464	6	0.20	6	0.00	6	0.00
U4	1000	484	2	0.02	2	0.01	2	0.01	U54	1000	443	5	0.23	6	0.00	6	0.00
U5	1000	474	2	0.01	2	0.01	2	0.00	U55	1000	456	5	0.17	6	0.00	5	0.00
U6	1000	465	2	0.07	3	0.00	3	0.00	U56	1000	457	6	0.20	6	0.00	6	0.00
U7	1000	487	2	0.02	2	0.01	2	0.01	U57	1000	437	5	0.17	5	0.00	5	0.00
U8	1000	474	2	0.02	2	0.01	2	0.01	U58	1000	435	6	0.18	6	0.00	6	0.00
U9	1000	473	2	0.02	2	0.01	2	0.00	U59	1000	464	6	0.20	6	0.00	6	0.00
U10	1000	468	2	0.02	2	0.01	2	0.00	U60	1000	447	6	0.20	6	0.00	6	0.00
U11	1000	452	3	0.06	3	0.00	3	0.01	U61	1000	433	7	0.24	7	0.00	7	0.00
U12	1000	466	3	0.05	3	0.01	3	0.01	U62	1000	460	6	0.27	7	0.01	6	0.01
U13	1000	467	3	0.06	3	0.01	3	0.01	U63	1000	439	7	0.24	7	0.00	7	0.00
U14	1000	461	3	0.06	3	0.00	3	0.01	U64	1000	453	6	0.24	7	0.01	6	0.00
U15	1000	470	3	0.07	3	0.00	3	0.01	U65	1000	448	7	0.26	7	0.00	7	0.00
U16	1000	465	3	0.06	3	0.01	3	0.01	U66	1000	477	6	0.23	6	0.01	6	0.01
U17	1000	477	3	0.06	3	0.00	3	0.00	U67	1000	444	6	0.32	6	0.00	6	0.00
U18	1000	466	3	0.08	3	0.01	3	0.00	U68	1000	448	6	0.23	6	0.00	6	0.00
U19	1000	467	3	0.06	3	0.01	3	0.01	U69	1000	473	6	0.24	6	0.01	6	0.00
U20	1000	461	3	0.06	3	0.00	3	0.01	U70	1000	439	6	0.21	6	0.00	6	0.00
U21	1000	445	4	0.09	4	0.01	4	0.01	U71	1000	439	7	0.30	8	0.00	7	0.00
U22	1000	478	4	0.10	4	0.00	4	0.00	U72	1000	456	7	0.31	7	0.00	7	0.01
U23	1000	476	4	0.09	4	0.00	4	0.00	U73	1000	444	7	0.28	7	0.00	7	0.00
U24	1000	471	3	0.08	3	0.00	3	0.00	U74	1000	438	7	0.31	7	0.00	7	0.00
U25	1000	454	4	0.09	4	0.01	4	0.00	U75	1000	465	6	277.14	7	0.01	7	0.00
U26	1000	449	4	0.09	4	0.00	4	0.01	U76	1000	439	7	0.28	7	0.00	7	0.00
U27	1000	456	4	0.09	4	0.00	4	0.01	U77	1000	431	7	0.32	8	0.01	7	0.00
U28	1000	462	3	0.07	3	0.00	3	0.00	U78	1000	456	6	0.30	6	0.00	6	0.01
U29	1000	477	3	0.89	4	0.00	4	0.00	U79	1000	453	8	0.33	8	0.00	8	0.00
U30	1000	457	4	0.09	4	0.01	4	0.00	U80	1000	437	7	0.31	7	0.01	7	0.00
U31	1000	446	4	0.09	4	0.00	4	0.00	U81	1000	409	7	7.01	8	0.00	8	0.00
U32	1000	470	5	0.13	5	0.00	5	0.00	U82	1000	451	8	0.39	8	0.00	8	0.01
U33	1000	463	5	0.12	5	0.00	5	0.00	U83	1000	452	7	8.23	8	0.00	8	0.00
U34	1000	449	4	0.10	4	0.00	4	0.00	U84	1000	447	7	0.41	7	0.00	7	0.00
U35	1000	454	4	0.09	4	0.00	4	0.00	U85	1000	441	7	10.37	8	0.01	8	0.00
U36	1000	455	4	0.11	4	0.00	4	0.00	U86	1000	451	7	7.31	8	0.00	8	0.01
U37	1000	444	4	0.10	5	0.01	4	0.00	U87	1000	452	7	0.36	7	0.00	7	0.00
U38	1000	445	4	0.11	4	0.00	4	0.00	U88	1000	452	7	0.44	7	0.00	7	0.00
U39	1000	472	4	0.10	4	0.00	4	0.00	U89	1000	437	8	0.40	8	0.00	8	0.01
U40	1000	442	5	0.13	5	0.00	5	0.00	U90	1000	430	7	7.96	8	0.00	8	0.00
U41	1000	439	5	0.15	5	0.00	5	0.00	U91	1000	441	8	7.60	9	0.00	9	0.00
U42	1000	475	5	0.15	5	0.00	5	0.00	U92	1000	467	8	51.12	9	0.00	9	0.00
U43	1000	476	5	0.15	5	0.00	5	0.00	U93	1000	450	8	0.43	8	0.00	8	0.00
U44	1000	452	4	0.11	4	0.00	4	0.00	U94	1000	449	8	0.55	9	0.00	8	0.00
U45	1000	469	4	2.66	5	0.00	5	0.00	U95	1000	462	8	0.55	8	0.00	8	0.00
U46	1000	456	5	0.15	5	0.00	5	0.00	U96	1000	446	8	7.94	9	0.00	9	0.00
U47	1000	478	5	0.16	5	0.00	5	0.00	U97	1000	475	8	0.49	8	0.01	8	0.00
U48	1000	465	4	2.92	5	0.00	5	0.00	U98	1000	462	8	0.54	8	0.00	8	0.00
U49	1000	436	5	0.14	5	0.00	5	0.00	U99	1000	444	8	0.45	8	0.00	8	0.00
U50	1000	454	5	0.15	5	0.00	5	0.00	U100	1000	438	8	0.44	8	0.01	8	0.00

Table 19.2 Heuristic algorithms comparison and computational behavior of formulation *Form_A*: Instance I

Inst	n	T	<i>Form_A</i>			Best fit			First fit								
			Obj	Time	T.L.	Obj	Time	T.L.	Obj	Time	T.L.						
11	2697	2688	12	T.L.	12	0.50	12	0.20	151	4948	768	26	T.L.	26	0.08	26	0.07
12	2825	2816	12	T.L.	12	0.58	12	0.23	152	5769	896	26	T.L.	26	0.14	26	0.08
13	2953	2944	12	T.L.	12	0.60	12	0.26	153	6721	1024	28	T.L.	28	0.47	28	0.17
14	3081	3072	12	T.L.	12	0.71	12	0.29	154	7382	1152	26	T.L.	26	0.84	26	0.30
15	3209	3200	12	T.L.	12	0.79	12	0.33	155	8266	1280	26	T.L.	26	1.15	26	0.33
16	3337	3328	12	T.L.	12	0.92	12	0.36	156	9002	1408	29	T.L.	29	1.74	29	0.48
17	3465	3456	11	T.L.	12	0.95	11	0.39	157	9865	1536	29	T.L.	29	2.19	29	0.69
18	3593	3584	11	T.L.	11	1.00	11	0.43	158	10,000	1664	29	T.L.	29	2.91	29	0.71
19	3721	3712	12	T.L.	12	1.18	12	0.47	159	10,000	1792	26	T.L.	26	2.55	26	0.76
110	3849	3840	12	T.L.	12	1.28	12	0.53	160	10,000	1920	27	T.L.	28	3.13	27	0.77
111	4480	2688	16	T.L.	16	1.19	16	0.45	161	2071	768	30	T.L.	30	0.03	30	0.02
112	4749	2816	15	T.L.	16	1.30	15	0.51	162	2422	896	28	T.L.	28	0.05	28	0.03
113	4887	2944	16	T.L.	16	1.49	16	0.56	163	2763	1024	27	T.L.	27	0.13	27	0.06
114	5153	3072	16	T.L.	16	1.67	16	0.63	164	3103	1152	28	T.L.	28	0.22	28	0.09
115	5298	3200	16	T.L.	16	1.76	16	0.69	165	3434	1280	27	T.L.	27	0.35	27	0.12
116	5547	3328	15	T.L.	15	1.83	15	0.75	166	3774	1408	27	T.L.	27	0.56	27	0.19
117	5745	3456	17	T.L.	17	2.27	17	0.85	167	4164	1536	26	T.L.	26	0.70	26	0.26
118	5929	3584	17	T.L.	17	2.41	17	0.90	168	4488	1664	27	T.L.	27	0.85	27	0.31
119	6208	3712	16	T.L.	16	2.58	16	0.98	169	4786	1792	27	T.L.	27	1.01	27	0.37
120	6462	3840	16	T.L.	16	2.77	16	1.03	170	5142	1920	29	T.L.	29	1.36	29	0.45
121	4972	2688	19	T.L.	19	1.53	19	0.58	171	2916	768	29	T.L.	29	0.06	29	0.04
122	5161	2816	22	T.L.	22	2.06	22	0.67	172	3424	896	28	T.L.	28	0.13	28	0.05
123	5423	2944	21	T.L.	21	2.24	21	0.74	173	3832	1024	28	T.L.	28	0.19	28	0.07
124	5658	3072	21	T.L.	21	2.45	21	0.82	174	4316	1152	28	T.L.	29	0.41	28	0.12
125	5875	3200	20	T.L.	20	2.68	20	0.89	175	4771	1280	27	T.L.	27	0.64	27	0.19
126	6112	3328	20	T.L.	20	2.77	20	0.98	176	5403	1408	30	T.L.	30	0.97	30	0.30
127	6380	3456	21	T.L.	21	2.99	21	1.09	177	5793	1536	29	T.L.	29	1.11	29	0.37
128	6582	3584	20	T.L.	20	3.06	20	1.17	178	6167	1664	27	T.L.	28	1.25	27	0.47
129	6817	3712	21	T.L.	21	3.48	21	1.27	179	6800	1792	28	T.L.	28	1.74	28	0.60
130	7026	3840	21	T.L.	21	3.69	21	1.38	180	7241	1920	28	T.L.	28	2.00	28	0.69
131	6322	2688	24	T.L.	24	2.32	24	0.88	181	5210	768	28	T.L.	28	0.14	28	0.06
132	6546	2816	23	T.L.	23	2.47	23	0.95	182	6057	896	26	T.L.	27	0.20	26	0.08
133	6895	2944	24	T.L.	24	2.97	24	1.14	183	6901	1024	29	T.L.	29	0.59	29	0.18
134	7209	3072	25	T.L.	25	3.50	25	1.22	184	7737	1152	29	T.L.	29	0.85	29	0.22
135	7484	3200	25	T.L.	25	3.47	25	1.27	185	8656	1280	28	T.L.	28	1.27	28	0.38
136	7784	3328	24	T.L.	24	4.13	24	1.51	186	9370	1408	28	T.L.	28	1.19	28	0.43
137	8089	3456	24	T.L.	24	4.55	24	1.67	187	10,000	1536	27	T.L.	27	2.11	27	0.63
138	8425	3584	23	T.L.	23	4.77	23	1.82	188	10,000	1664	27	T.L.	27	2.20	27	0.68
139	8650	3712	24	T.L.	24	5.50	24	1.96	189	10,000	1792	29	T.L.	29	2.79	29	0.70
140	8977	3840	23	T.L.	23	5.53	23	2.18	190	10,000	1920	29	T.L.	29	2.95	29	0.75
141	2071	768	26	T.L.	26	0.03	26	0.02	191	3117	768	31	T.L.	30	0.25	31	0.10
142	2422	896	26	T.L.	26	0.06	26	0.03	192	3594	896	33	T.L.	33	0.13	33	0.06
143	2756	1024	27	T.L.	27	0.14	27	0.06	193	4176	1024	32	T.L.	32	0.27	32	0.10
144	3104	1152	27	T.L.	27	0.22	27	0.08	194	4671	1152	32	T.L.	32	0.40	32	0.13
145	3433	1280	27	T.L.	27	0.36	27	0.13	195	5209	1280	31	T.L.	31	0.71	31	0.22
146	3789	1408	28	T.L.	28	0.60	28	0.20	196	5628	1408	32	T.L.	32	0.99	32	0.33
147	4154	1536	29	T.L.	29	0.67	29	0.24	197	6215	1536	31	T.L.	31	1.23	31	0.43
148	4476	1664	27	T.L.	27	0.85	27	0.31	198	6730	1664	33	T.L.	33	1.73	33	0.56
149	4797	1792	28	T.L.	27	0.99	28	0.37	199	7172	1792	32	T.L.	31	1.96	32	0.67
150	5129	1920	27	T.L.	27	1.26	27	0.47	1100	7709	1920	31	T.L.	32	2.30	31	0.90

Time limit (T.L.) = 5 min

Table 19.3 Heuristic algorithms comparison and computational behavior of formulation *Form_A*

Group	n	T	<i>Form_A</i>			Best fit			First fit		
			Obj	Time	Exit Gap	Obj	Time	Gap	Obj	Time	Gap
U inst.											
I	1000	473	2.00	0.02	0.00	2.10	0.01	5.00	2.10	0.00	5.00
II	1000	465	3.00	0.06	0.00	3.00	0.01	0.00	3.00	0.01	0.00
III	1000	462	3.70	0.17	0.00	3.80	0.00	3.33	3.80	0.00	3.33
IV	1000	454	4.30	0.11	0.00	4.40	0.00	2.50	4.30	0.00	0.00
V	1000	460	4.70	0.67	0.00	4.90	0.00	5.00	4.90	0.00	5.00
VI	1000	451	5.60	0.19	0.00	5.90	0.00	6.00	5.70	0.00	2.00
VII	1000	451	6.30	0.25	0.00	6.50	0.00	3.33	6.30	0.00	0.00
VIII	1000	445	6.90	27.99	0.00	7.20	0.00	4.52	7.00	0.00	1.67
IX	1000	442	7.20	4.29	0.00	7.70	0.00	7.14	7.70	0.00	7.14
X	1000	453	8.00	7.01	0.00	8.40	0.00	5.00	8.30	0.00	3.75
I inst.											
I	3273	3264	11.80	T.L.	6.97	11.90	0.85	0.91	11.80	0.35	0.00
II	5445	3264	16.00	T.L.	28.39	16.10	1.93	0.67	16.00	0.74	0.00
III	6000	3264	20.60	T.L.	–	20.60	2.70	0.00	20.60	0.96	0.00
IV	7638	3264	23.90	T.L.	–	23.90	3.92	0.00	23.90	1.46	0.00
V	3613	1344	27.20	T.L.	–	27.10	0.52	0.00	27.20	0.19	0.37
VI	8195	1344	27.20	T.L.	–	27.30	1.52	0.37	27.20	0.44	0.00
VII	3614	1344	27.60	T.L.	–	27.60	0.53	0.00	27.60	0.19	0.00
VIII	5066	1344	28.20	T.L.	–	28.40	0.85	0.73	28.20	0.29	0.00
IX	8393	1344	28.00	T.L.	–	28.10	1.43	0.38	28.00	0.41	0.00
X	5422	1344	31.80	T.L.	–	31.70	1.00	0.32	31.80	0.35	0.66

Time limit (T.L.) = 5 min

solved to proven optimality and “–” means no dual bound). The column *Gap* reports the average ratio $\frac{z^A - z^*}{z^*}$, where z^* and z^A are the objective function values of the best known solution and that of the solution given by the corresponding algorithm.

We can see that for both sets of instances, the best fit algorithm and the first fit algorithm have the same results in more than 90% of the cases, while the first fit algorithm is computationally more efficient. Hence, we choose the first fit algorithm to provide the initial solutions for the other heuristic algorithms, including providing the initial solution for *Form_A*. We also point out that no I instance can be solved to proven optimality by the compact model *Form_A* within the time limit, whereas all the U instances are optimally solved.

We now test the column generation based heuristic algorithm introduced in Sect. 19.3.1, together we also compare the different lower bounds of the TBPP discussed in this paper, i.e., the LP relaxation of *Form_A*, *Form_B* and the bound based on the BPP in Property 19.1. The initial solution of the algorithm is provided by the first fit algorithm. Since the U instances are easy to solve, we only consider the I instances. Moreover, I instances are computational hard for *Form_B* as well, we only report results for the subset of instances of which we are able to solve its LP relaxation within 1 h via column generation. The results are presented in Table 19.4. For formulation *Form_A*, in column z_{lp} and *Time*, we report the objective function value of the LP relaxation and the computational time of CPLEX. The lower bound based on the BPP is given in column LB_{BPP} . Computing the best LB_{BPP} requires the solution of |T| BPPs and accordingly it may require a very long computational time. We decide instead to solve only 100 BPPs for randomly sampled time steps

Table 19.4 The column generation based heuristic algorithm and lower bounds of the TBPP

Instance	<i>Form_A</i>		<i>LB_{BPP}</i>	<i>Form_B</i>				
	z_{lp}	Time		z_{lp}	Pricing Time	Time	Cols	Obj
I40	18.35	268.31	20	23	321.71	1034.28	789	23
I44	21.53	34.27	26	26	1862.41	2639.48	1631	27
I45	22.18	46.84	24	27	0.48	0.53	81	27
I61	20.69	300.34	26	26	2001.04	3011.65	1924	40
I64	21.43	41.73	26	27	2237.6	2551.92	1091	28
I71	23.41	130.53	28	28	482.14	494.61	514	29
I72	23.07	21.57	26	27	1676.97	1823.78	985	28
I73	22.96	28.75	28	27.5	794.43	831.12	593	28
I75	23.66	42.71	24	27	8.56	13.33	284	27
I76	23.61	44.17	27	29	885.51	948.59	736	30
I77	23.2	73.38	26	29	32.61	38.47	530	29
I78	22.4	76.32	27	27	1035.21	1295.63	566	27
I82	22.7	26.08	26	26	1888.26	2757.68	904	26
I83	24.46	27.98	28	28	480	564.07	896	29
I86	23.2	64.88	27	28	141.08	161.43	742	28
I88	22.62	65.42	26	27	12.17	12.7	135	27
I89	23.79	73.45	29	29	14.88	23.91	717	29
I90	24.53	74.82	28	29	142.55	229.19	1512	29
I93	26.58	49.61	31	31	1163.19	1235.46	752	32
I95	25.14	75.06	31	31	53.06	81.47	417	31
I96	25.55	88.64	32	32	56.23	88.91	477	32
I98	27.58	131.05	30	33	17.17	26.27	378	33
I99	27.06	142.98	29	31	1322.96	1438.21	664	32

in order to achieve a good compromise between computational time and quality of the bounds. Finally, for *Form_B*, column *Obj* and z_{lp} give the objective function value obtained by the heuristic algorithm proposed in Sect. 19.3.1 and that of the LP relaxation. The pricing time to solve the sub problem TKP during column generation and the total computational time of solving the LP relaxation are reported in column *Pricing Time* and *Time* respectively. The time for solving the final MIP of the column generation heuristic is on average very low except in some rare cases in which this phase can take up to 1 min. In our experiments, the TKP sub problems are solved by CPLEX using Formulation (19.11)–(19.13). Table 19.4 also presents the number of columns generated during the entire process.

It is worth mentioning that the computation of the LP relaxation of *Form_A* is surprisingly high while it can be alternatively computed in negligible time using the formula (19.6) presented in Property 19.2. In any case, this bound is very poor compared to other ones. We observe that in all cases, the extensive formulation returns better lower bounds. Also, the lower bounds based on the BPP are strictly better than the ones provided by *Form_A* in all cases except one. Finally, thanks to our column generation algorithm, we can prove the optimality for 14 instances out of 23 instances considered. In Table 19.4, we highlight objective function values of LP relaxation and that obtained by the heuristic algorithm based on *Form_B* for these 14 instances. Many of the I instances still remain open and we hope this paper may stimulate further research on the TBPP.

19.5 Conclusion

In this paper, we study an extension of the bin packing problem called TBPP and present a compact MIP formulation and an extensive MIP formulation. Also, greedy-type heuristic algorithms are developed, while an MIP solver together with the compact formulation is used as a heuristic given time limits. A heuristic algorithm based on the extensive formulation together with the column generation technique is presented. Computational experiments are discussed, showing the strength of the extensive formulation and the potentiality of column generation based algorithms applied to the TBPP.

For future research, a pseudo-polynomial size formulation could be developed. Also, other heuristic algorithms based on column generation can be studied, such as combining diving strategies and variable fixing.

Acknowledgements The authors want to thank Professor Paolo Toth and Professor Ivana Ljubic for stimulating discussions on the topic and for their contributions.

References

1. M. Bartlett, A.M. Frisch, Y. Hamadi, I. Miguel, S.A. Tarim, C. Unsworth, The temporal knapsack problem and its solution, in *Proceedings of the 2nd International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR 2005)* (2005), pp. 34–48
2. P. Bonsma, J. Schulz, A. Wiese, A constant-factor approximation algorithm for unsplittable flow on paths. *SIAM J. Comput.* **43**(2), 767–799 (2014)
3. A. Caprara, F. Furini, E. Malaguti, Uncommon Dantzig-Wolfe reformulation for the temporal knapsack problem. *INFORMS J. Comput.* **25**(3), 560–571 (2013)
4. A. Ceselli, G. Righini, M. Salani, A column generation algorithm for a vehicle routing problem with economies of scale and additional constraints. *Transp. Sci.* **43**(1), 56–69 (2009)
5. E.G. Coffman Jr., M.R. Garey, D.S. Johnson, Approximation algorithms for bin-packing—an updated survey, in *Algorithm Design for Computer System Design* (Springer, Vienna, 1984), pp. 49–106
6. E.G. Coffman Jr., J. Csirik, G. Galambos, S. Martello, D. Vigo, Bin packing approximation algorithms: survey and classification, in *Handbook of Combinatorial Optimization*, ed. by P.M. Pardalos, D. Du, R.L. Graham (Springer, New York, 2013), pp. 455–531
7. F. Furini, E. Malaguti, R. Medina Durán, A. Persiani, P. Toth, A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size. *Eur. J. Oper. Res.* **218**(1), 251–260 (2012)
8. P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting-stock problem. *Oper. Res.* **9**(6), 849–859 (1961)
9. P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting stock problem - part II. *Oper. Res.* **11**(6), 863–888 (1963)
10. IBM-CPLEX, <http://www-01.ibm.com/software/integration/optimization/-cplex-optimizer/>
11. D. Johnson, Near-optimal bin packing algorithms. Ph.D. dissertation, Department of Mathematics, M.I.T., Cambridge, MA (1973)
12. S. Lavoie, M. Minoux, E. Odier, A new approach for crew pairing problems by column generation with an application to air transportation. *Eur. J. Oper. Res.* **35**(1), 45–58 (1988)
13. S. Martello, P. Toth, *Knapsack Problems: Algorithms and Computer Implementations* (Wiley, New York, 1990)

Chapter 20

A Fast Reoptimization Approach for the Dynamic Technician Routing and Scheduling Problem

V. Pillac, C. Guéret, and A.L. Medaglia

Abstract The Technician Routing and Scheduling Problem (TRSP) consists in routing staff to serve requests for service, taking into account time windows, skills, tools, and spare parts. Typical applications include maintenance operations and staff routing in telecoms, public utilities, and in the health care industry. In this paper we tackle the Dynamic TRSP (D-TRSP) in which new requests appear over time. We propose a fast reoptimization approach based on a parallel Adaptive Large Neighborhood Search (RpALNS) able to achieve state-of-the-art results on the Dynamic Vehicle Routing Problem with Time Windows. In addition, we solve a set of randomly generated D-TRSP instances and discuss the potential gains with respect to a heuristic modeling a human dispatcher solution.

Keywords Dynamic Vehicle Routing • Technician Routing and Scheduling • Parallel Adaptive Large Neighborhood Search

V. Pillac

Ecole des Mines de Nantes, IRCCyN UMR 6597, Nantes, France
Centro para la Optimizacion y Probabilidad Aplicada (COPA), Universidad de los Andes,
Bogota, Colombia
e-mail: victor.pillac@gmail.com

C. Guéret (✉)

LARIS, Université d'Angers, Angers, France
e-mail: christelle.gueret@univ-angers.fr

A.L. Medaglia

Centro para la Optimizacion y Probabilidad Aplicada (COPA), Universidad de los Andes,
Bogota, Colombia
e-mail: amedagli@uniandes.edu.co

20.1 Introduction

The Technician Routing and Scheduling Problem (TRSP) [22] deals with a limited crew of technicians \mathcal{H} that serves a set of requests \mathcal{R} . The TRSP can be seen as an extension of the Vehicle Routing Problem with Time Windows (VRPTW), where technicians play the role of vehicles and requests are made by clients. In the TRSP, each technician has a set of skills, tools, and spare parts, while requests require a subset of each. The problem is then to design a set of routes of minimal duration such that each request is visited exactly once, within its time window, by a technician with the required skills, tools, and spare parts. The TRSP naturally arises in a wide range of applications, including telecommunications, public utilities, and maintenance operations.

This problem introduces compatibility constraints between technicians and requests. While skills are intrinsic attributes, technicians may carry different tools and spare parts over the planning horizon. Technicians usually start their route from their home with an initial set of tools and spare parts that allows them to serve an initial set of requests. They can also replenish their tools and spare parts at a central depot at any time to serve more requests. Tools can be seen as renewable resources, while spare parts are non renewable and consumed once the technician serves a request.

Figure 20.1 illustrates an instance of the TRSP with two technicians and six requests. Technician A has one skill, while B has two. Technician A starts their route at home (gray diamond) with a hammer and a screwdriver, then serves requests 1, 2, and 3, before returning home. Technician B also departs from their home, serves 4, then goes to the central depot (black square) to pick up a drill that allows them to serve request 6 after serving request 5. Note that although request 5 is close to the route of technician A, only technician B can serve it due to skill constraints.

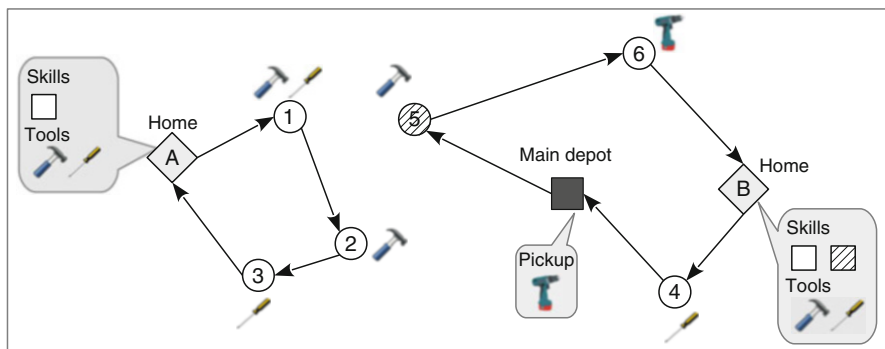


Fig. 20.1 Example of a technician routing and scheduling problem with two technicians, three tools, and two skills

In this work, we introduce a dynamic variant of the problem, namely the D-TRSP, in which new requests appear while the technicians are executing their routes. In addition to designing a set of routes at the beginning of the day, two types of decisions have to be taken in real time. First, whenever a technician finishes serving

a request, it must be decided what will be the next request to serve. Second, whenever a request appears, the algorithm must decide whether it is possible or desirable to accept it or not. If not, the request is said to be *rejected*, which leads to a cost penalty corresponding to the cost of outsourcing or postponing the request. This problem is motivated by a practical application in which no stochastic information is available to anticipate new requests, while decisions need to be made in limited time to provide an appropriate level of service to customers. Therefore, we propose a fast reoptimization approach that focuses on providing good solutions in limited time. Finally, we compare two distinct objective functions, the first minimizing the duration of routes, and the second the traveled distance.

The rest of this chapter is organized as follows. Section 20.2 reviews the literature on related problems and approaches. Section 20.3 describes a parallel adaptive Large Neighborhood Search (pALNS), Sect. 20.4 introduces a fast reoptimization approach based on pALNS, Sect. 20.5 presents computational results for the proposed approach, and finally Sect. 20.6 concludes this paper and draws directions for future research.

20.2 Literature Review

Despite its numerous practical applications and its challenging features, static technician routing and scheduling problems have received limited attention until recently, and to the best of our knowledge, no study simultaneously considers skills, tools, and spare parts. For instance, Xu and Chiu [37] studied the Field Technician Scheduling Problem (FTSP) seen as a variant of the VRPTW, in which the objective is to maximize the number of requests served while accounting for skill constraints, request priorities, multiple depots, and overtime. The authors describe a mixed integer formulation and develop three heuristics including a GRASP algorithm. Similarly, Weigel and Cao [36] present a software solution developed for Sears, a US retailer that serves its customers with home delivery and on-site technical assistance. The proposed solution works by first assigning technicians to requests, and then optimizing technician routes individually. Tsang and Voudouris [33] studied the technician workforce scheduling problem faced by British Telecom. Their study does not consider skill constraints, but uses a proficiency factor that reduces the service time depending on the technician experience. They propose a Fast Local Search and a Guided Local Search to solve this problem. Borenstein et al. [7] extended this problem accounting for dynamic requests and skill compatibility constraints. They cluster the requests using a k -means algorithm followed by a heuristic that assigns technicians to areas. Finally, they propose a rule-based system that assigns and sequences the requests. They conclude their study by assessing the impact of soft clustering and show that it can increase system performance under certain assumptions.

Maintenance operations planning is a problem closely related to the TRSP. Blakeley et al. [6] present the optimization of periodic maintenance operations for Schindler Elevator Corporation in North America, a company that manufactures,

installs, and maintains elevators and escalators. The problem faced by this company consists in designing a set of routes for technicians to perform periodic maintenance and repairs taking into account travel times, working regulations, and skill constraints. A similar application was studied by Tang et al. [31] who formulate the problem as a multi-period maximum collection problem in which time-dependent rewards are granted for the completion of a request. This approach allows the modeling of soft constraints such as the desirability of performing a task in a given day (*job-to-time penalties*). The authors propose a Tabu Search (TS) algorithm that yields near-optimal solutions on real instances in reasonable time.

In 2007 the French Operations Research Society (ROADEF) organized a challenge based on a problem submitted by France Telecom. The problem consists in finding a schedule for technicians to execute a set of tasks on a multiple-day horizon. Each task requires one or more skills with different minimum proficiency levels, while technicians can have multiple skills with a given proficiency. An important aspect is the creation of teams that work together during one day, combining the skills of their members, and the possibility to outsource the execution of a task. However, this problem ignores the routing aspects. Cordeau et al. [11] proposed a mathematical model and an Adaptive Large Neighborhood Search (ALNS), while Hashimoto et al. [14] proposed a Greedy Randomized Adaptive Search procedure (GRASP) approach to tackle this problem.

Work regulation is an important aspect of technician routing and scheduling. Tricoire [32] presents a technician routing problem faced by Veolia, a water distribution and treatment company. In this application, technicians have the skills to perform all requests that are divided in two categories: user requested interventions and company scheduled visits. As new requests appear on a daily basis, the routing of technicians is performed on a rolling horizon, taking into account work regulations and customer service standards. In his work, the main contributions are a column generation approach and a memetic algorithm [8]. His approaches take advantage of partial solutions from previous plans in the rolling horizon framework to reduce computational times.

A number of technological advances have led to a renewed interest in dynamic vehicle routing problems, leading to the development of new optimization approaches. Pillac et al. [21] classify dynamic routing problems in two categories: *deterministic* and *stochastic*. In both cases the information available to the stakeholder changes over time. In a stochastic setting, data is available on the dynamically revealed information in the form of known probability distributions, while in deterministic problems, changes are simply unpredictable. The present work falls in the dynamic and deterministic category.

Dynamic and deterministic problems are often tackled with approaches either based on *periodic reoptimization* or *continuous reoptimization* [21]. Periodic reoptimization approaches start at the beginning of the day by producing an initial set of routes that are communicated to the vehicles. As the available information is updated along the day, or at given intervals of time, an optimization is performed using the currently available information to update the routing. Such approaches can be based on algorithms developed for static problems and are therefore relatively easy to implement, however, they may introduce delays between the information

update and the routing plan. Examples of periodic reoptimization include the Ant Colony Systems proposed by Montemanni et al. [20] to solve the Dynamic VRP (D-VRP). A novel feature of their approach is the use of the pheromone trace to transfer characteristics of a good solution between reoptimizations. Other heuristic approaches, such as Tabu Search (TS), have also been used to tackle the Dynamic Pickup and Delivery Problem (D-PDP) [9, 2] and the Dynamic Dial-a-Ride Problem (D-DARP) [1, 3].

Continuous reoptimization approaches run throughout the day and are generally based on an adaptive memory [30] that stores alternative solutions. The adaptive memory is then used to react to changes in the available information, thus avoiding a complete reoptimization of the problem. Gendreau et al. [12] developed a parallel TS with adaptive memory to tackle a Dynamic VRPTW (D-VRPTW), that was later applied to the D-VRP [16, 17]. Bent and Van Hentenryck [4] generalized this framework and introduced the Multiple Plan Approach (MPA) to tackle the D-VRPTW. Following a different approach, Benyahia and Potvin [5] studied the Dynamic Pickup and Delivery Problem (D-PDP) and proposed a Genetic Algorithm (GA) that models the decision process of a human dispatcher. More recently, GAs were also used for the same problem [13, 10] and for the D-VRP [34].

To the best of our knowledge, no work considers simultaneously skills, tools, spare parts, and dynamically arriving requests, four key and practical components of technician routing and scheduling. The present work addresses this aspect and proposes an optimization approach for the dynamic version of the problem, denoted D-TRSP, in which new requests arrive during the execution of the routes.

20.3 The Parallel Adaptive Large Neighborhood Search

The proposed approach is based on a parallel Adaptive Large Neighborhood Search (pALNS) algorithm which is used to compute an initial solution, and then, to reoptimize the solution whenever a new request arrives. In this section we present the original Adaptive Large Neighborhood Search (ALNS) algorithm and introduce an efficient parallelization scheme.

The ALNS algorithm, originally proposed by Pisinger and Ropke [23], is an extension of the Large Neighborhood Search (LNS) algorithm [28]. LNS works by successively destroying (removing requests) and repairing (inserting requests back) a current solution, using *destroy* and *repair operators*. ALNS adds an adaptive layer that randomly selects operators depending on their past performance, automatically fitting the algorithm to the instance at hand. We refer the interested reader to Pisinger and Ropke [24] for a detailed description of LNS, ALNS, and related methods.

Algorithm 1 presents the outline of the ALNS approach. ALNS starts with an initial solution Π_0 . Then for I iterations, the algorithm selects destroy and repair operators with a roulette wheel that reflects their past performance (line 4). Destroy operators remove a subset of requests from the current solution, while repair operators reinsert them using heuristics that are known to perform well on the problem at hand (line 5). The resulting new solution is conditionally accepted as current

Algorithm 1 Adaptive Large Neighborhood Search (ALNS) algorithm

Input: Π_0 initial solution, z evaluation function, Θ^-/Θ^+ set of destroy/repair operators, I number of iterations

Output: Π^* the best solution found

```

1:  $\Pi^* \leftarrow \Pi_0$  {Initialize best solution}
2:  $\Pi \leftarrow \Pi_0$  {Initialize current solution}
3: for  $I$  iterations do
4:    $d \leftarrow \text{select}(\Theta^-); r \leftarrow \text{select}(\Theta^+)$  {Select destroy/repair}
5:    $\Pi' \leftarrow r(d(\Pi))$  {Generate a neighbor}
6:   if  $\text{accept}(\Pi', \Pi)$  then
7:      $\Pi \leftarrow \Pi'$  {Update current solution}
8:   end if
9:   if  $z(\Pi') < z(\Pi^*)$  then
10:     $\Pi^* \leftarrow \Pi'$  {Update best solution}
11:   end if
12:    $\text{updateScore}(d, r, \Pi')$  {Update scores}
13: end for
14:
15: return  $\Pi^*$ 

```

solution according to a simulated annealing criterion (line 6). At the end of each iteration, the scores of the destroy and repair operators are updated depending on the solution they generated (line 12).

The Parallel Adaptive Large Neighborhood Search (pALNS) was briefly introduced by the authors in Pillac et al. [22], the remaining of this section details its components. It is an extension of the Adaptive Large Neighborhood Search (ALNS) algorithm that includes a novel parallelization scheme that efficiently spreads the computational effort among independent processors.

Algorithm 2 presents the outline of pALNS. The algorithm maintains a pool \mathcal{P} of N promising solutions that are optimized in K subprocesses (note that $N \geq K$). For each *master* iteration, a subset of K promising solutions is selected randomly (line 2) and distributed among independent subprocesses. Each subprocess performs I^p ALNS iterations (lines 3–14) by destroying and repairing the current solution Π^p as in the original ALNS algorithm. The final current solution of each subprocess is added to the pool of promising solutions (line 13) and a filtering procedure ensures that the pool contains at most N solutions, including the best solution found so far (line 15). The algorithm stops after I^m master iterations, which corresponds to $I = I^m \times I^p$ ALNS iterations. Note that the implementation of pALNS ensures that no synchronization is required between subprocesses to avoid deadlocks. The following paragraphs present in more detail the different components of the algorithm.

20.3.1 Destroy

Destroy operators remove a random fraction $\xi \in [\xi_{min}, \xi_{max}]$ of the requests from the current solution. We denote \mathcal{R} the set of requests served in the solution, and \mathcal{U}

Algorithm 2 Parallel Adaptive Large Neighborhood Search (pALNS) algorithm

Input: \mathcal{P} pool of initial solutions, z evaluation function, Θ^-/Θ^+ set of destroy/repair operators, N maximum size of the solution pool, K number of subprocesses, I^m number of master iterations, I^p number of iterations performed in parallel.

Output: Π^* , the best solution found

```

1: for  $I^m$  iterations do
2:    $\mathcal{P}' \leftarrow \text{selectSubset}(\mathcal{P}, K)$  {Select a subset of  $K$  solutions}
3:   for All  $\Pi$  in  $\mathcal{P}'$  do
4:      $\Pi^p \leftarrow \Pi$  {Current solution for this subprocess}
5:     for  $I^p$  iterations do
6:        $d \leftarrow \text{select}(\Theta^-); r \leftarrow \text{select}(\Theta^+)$  {Select destroy/repair}
7:        $\Pi' \leftarrow r(\Pi^p)$  {Destroy and repair current solution}
8:       if accept( $\Pi', \Pi^p$ ) then
9:          $\Pi^p \leftarrow \Pi'$  { $\Pi'$  is accepted as current solution}
10:      end if
11:      updateScore( $d, r, \Pi'$ ) {Update  $d$  and  $r$  scores}
12:    end for
13:     $\mathcal{P} \leftarrow \mathcal{P} \cup \{\Pi^p\}$  {Add  $\Pi^p$  to the pool  $\mathcal{P}$ }
14:  end for
15:   $\mathcal{P} \leftarrow \text{retain}(\mathcal{P}, N)$  {Retain at most  $N$  solutions from the pool  $\mathcal{P}$ }
16: end for
17:
18: return  $\Pi^* = \arg \min_{\Pi \in \mathcal{P}} \{z(\Pi)\}$ 

```

the set of requests that are not served. We used three destroy operators originally proposed by Pisinger and Ropke [23]: *random*, *related*, and *critical*.

The random destroy operator selects requests randomly and removes them from their actual routes.

The related destroy operator attempts to remove requests that share some characteristics. Let the *relatedness* r_{ij} of requests i and j be a measure of how related two requests are (the lower the r_{ij} , the more related i and j). The procedure starts by randomly removing a *seed* request i ($\mathcal{U} = \{i\}$), then it iteratively selects a request $i \in \mathcal{U}$, and removes the most related request j^* :

$$j^* = \arg \min_{j \in \mathcal{R}} \{r_{ij}\} \quad (20.1)$$

There are different ways to measure relatedness. We propose a new metric that can be precalculated, namely a-priori relatedness, that does not depend on the actual position of requests in routes:

$$r_{ij}^s = \left(1 + \frac{c_{ij}}{M_c}\right)^{\theta_c} \left(1 + \frac{|b_i - b_j|}{M_t}\right)^{\theta_t} \left(2 - \frac{|\mathcal{K}_i \cap \mathcal{K}_j|}{\min\{|\mathcal{K}_i|, |\mathcal{K}_j|\}}\right)^{\theta_s} \quad (20.2)$$

where M_c and M_t are scaling constants, and θ_c , θ_t , and θ_s are factors that define the weight given to each component. The first component measures the geographic distance between two requests (c_{ij}); the second component is the difference of due dates b_i and b_j ; and the third component measures the number of technicians that

can serve both requests, modeled by the intersection of \mathcal{K}_i and \mathcal{K}_j , the sets of technicians that can serve request i and j , respectively.

In addition, we use time-oriented relatedness [23] that measures the difference between the current times of service A_i and A_j of requests i and j :

$$r_{ij}^t = |A_i - A_j| \quad (20.3)$$

Finally, critical destroy removes the request i^* such that the cost of the resulting solution is minimal:

$$i^* = \arg \max_{i \in \mathcal{R}} \{c_{i-1, i+1} - c_{i-1, i} - c_{i, i+1}\} \quad (20.4)$$

where $i-1$ and $i+1$ are the predecessor and successor of i .

In practice, related and critical operators are randomized and the $\lfloor y^p |\mathcal{R}| \rfloor$ -th best request is selected, where y is a random number within $[0, 1)$ and $p \geq 1$ is a parameter that controls the level of randomness (the lower the p , the more randomness is introduced).

20.3.2 Repair

Repair operators attempt to insert requests that are currently unserved. Our implementation is based on *regret-q heuristics* [25]: at each iteration the algorithm inserts (at the best position) the request with the lowest *regret* value. The regret-q value r_i^q of request i is a measure of how desirable it is to insert i in the current iteration assuming that the best insertion will no longer be feasible in the next iteration. It is defined as:

$$r_i^q = \sum_{h=2}^q \left(\Delta z_i^h - \Delta z_i^1 \right) \quad (20.5)$$

where Δz_i^q is the cost of the q -th best insertion of request $i \in \mathcal{U}$. Note that ties are broken by selecting the request with the lowest Δz_i^1 value, and regret-1 corresponds to the classical best insertion heuristic. We used three regret levels: regret-1, regret-2, and regret-3.

20.3.3 Adaptive Layer

Every iteration, pALNS selects a destroy and a repair operator using a selection roulette, such that operator $\theta \in \Theta^*$ is selected with probability w_θ , where Θ^* is either the set of destroy (Θ^-) or repair (Θ^+) operators. Probabilities are initialized with value $\frac{1}{|\Theta^*|}$, and then updated every l iterations (a *segment*) as follows:

$$w_\theta \leftarrow (1 - \rho)w_\theta + \rho \frac{s_\theta}{\sum_{\theta \in \Theta} s_\theta} \quad (20.6)$$

where $\rho \in [0, 1]$ is the *reaction factor* which defines how quickly probabilities are adjusted, and s_θ is the *score* of operator θ in the last l iterations. The scores s_θ are reset to 0 every l iterations and updated at the end of each iteration depending on the new solution: a score of σ_1 is granted for a new best solution, σ_2 for an improving solution, σ_3 for a non-improving but accepted solution, and σ_4 for a rejected solution. It is worth noting that in contrast with the adaptive scheme originally proposed by Pisinger and Ropke [23], this formula ensures that $\sum_{\theta \in \Theta} w_\theta = 1$ at all time, which makes it easier to interpret as the relative weight of each operator.

20.3.4 Objective Function

The initial solution or the solution resulting from the destroy operator can leave some requests unserved ($\mathcal{U} \neq \emptyset$). Therefore, we need to be able to evaluate a partial solution Π' to account for the unserved requests. Let z be one of the two objective functions considered in this paper, and Π_0 an initial solution. Pisinger and Ropke [23] define the cost of partial solution Π' as follows:

$$z_\phi(\Pi') = z(\Pi') + \phi |\mathcal{U}| z(\Pi_0) \quad (20.7)$$

where ϕ is a parameter that controls the unserved request penalty. Note that in a dynamic context, the penalty can be interpreted as an outsourcing cost for rejected requests.

20.3.5 Acceptance Criterion

As in the original ALNS, the pALNS algorithm relies on a simulated annealing acceptance criterion which accepts a new solution Π' with probability $e^{\frac{z(\Pi) - z(\Pi')}{T}}$, where T is the *temperature* parameter. The temperature is initialized with the value T_0 and it is reduced at each iteration by a *cooling factor* c . The two parameters T_0 and c are set depending on the initial solution and the target number of iterations [27]. Given an initial solution Π_0 , T_0 is defined such that a solution with value $(1 + w)z(\Pi_0)$ is accepted with probability p , and c is set such that the temperature after n iterations is equal to αT_0 .

20.3.6 Computation of an Initial Solution

The pALNS algorithm requires an initial solution which is computed with a regret-3 constructive heuristic: starting with empty routes for each technician, the algorithm iteratively inserts the request with the lowest regret value as described in Sect. 20.3.2.

20.3.7 Solution Pool

The solution pool acts as a shared memory and allows subprocesses to collaborate efficiently. In the original algorithm, the simulated annealing acceptance criterion results in a search scheme that starts from a diversification phase, in which poor solutions may be accepted as current solutions, and progressively switch to an intensification phase, in which only improving solutions are accepted. The use of a solution pool that contains the N best solutions found so far tends to break this scheme, as poor solutions may never be kept in the pool and will not be exploited properly. To overcome this limitation we propose to maintain a pool of *diverse* solutions that are promising in terms of cost.

This is achieved by the `retain` method (line 15) which ensures that \mathcal{P} contains at most N solutions: if $|\mathcal{P}| > N$ then the method retains the N best solutions according to the fitness function f :

$$f(\Pi) = (1 - \lambda)\text{rank}_z(\Pi) + \lambda\text{rank}_d(\Pi) \quad (20.8)$$

where λ is a weight between 0 and 1, $\text{rank}_z(\Pi)$ is the rank of solution Π according to its objective value, and $\text{rank}_d(\Pi)$ is the rank of Π according to its average *broken-pairs distance* [26] relative to the other solutions from \mathcal{P} . The broken pairs distance counts the number of arcs that differ between two solutions. This fitness function is inspired by the *biased fitness* introduced by Vidal et al. [35] in a genetic algorithm with diversity management. The weight λ can either be fixed a-priori, or adjusted throughout the search to switch from diversification ($\lambda = 1$) to intensification ($\lambda = 0$). Note that we ensure that \mathcal{P} always contains the best solution found so far.

20.4 Parallel Reoptimization Approach

Figure 20.2 illustrates the proposed reoptimization approach, namely RpALNS. The algorithm starts by producing an initial solution S_0 by using a constructive heuristic coupled with the pALNS described in the previous section. Then each time a new request appears, it fixes the currently executed portion of the routes, and re-runs the pALNS for a limited number of iterations to produce an updated solution S'_t . If pALNS is able to insert the new request, then it is *accepted* and S'_t becomes

the new current solution, otherwise the request is *rejected* and S_t remains as the current solution.

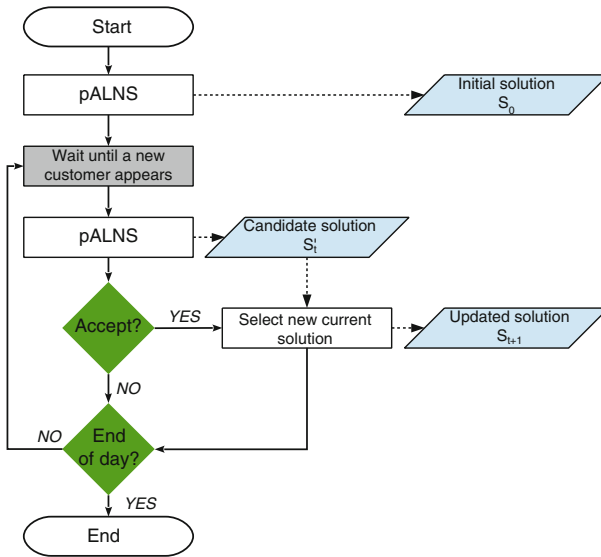


Fig. 20.2 Overview of the proposed reoptimization approach

It is important to note that the immediate commitment of idle technicians to requests may lead to difficulties when new requests appear. Figure 20.3 illustrates this with a single technician. Suppose that at time t a technician is assigned to a request i , if the technician is dispatched immediately to i (upper left time line), it will travel to i then wait at its destination until the start of the time window (black brackets). On the other hand, if a waiting strategy is used (lower left time line), the technician will remain idle until the latest moment such that it will not wait at i . If at time $t + 1$ a new request j appears, in the first case j cannot be served as the technician is already waiting at i , while in the second case a visit to j can be inserted right before i . As a consequence, technicians are considered to remain idle at their current location until the latest departure time.

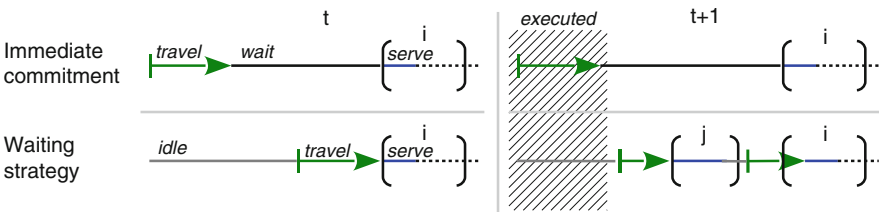


Fig. 20.3 Illustration of waiting vs. immediate commitment strategies

20.5 Computational Results

In this section we first validate the proposed reoptimization approach on a well known benchmark and present computational results for randomly generated instances of the D-TRSP.

20.5.1 Experimental Setting

The experiments were run on a quad-core desktop computer.¹ Table 20.1 presents the detail parameter setting used in the pALNS algorithm. The number of parallel iterations and the maximum size of the pool where selected after running experiments with values $I^P \in \{10, 50, 100, 500, 1000\}$ and $N \in \{1, 5, 10, 20, 30, 40, 50\}$. We also tested two schemes for the solution pool, the first with a fixed value of 0.5 for λ , the second using an adaptive scheme starting with $\lambda = 0.5$ and decreasing its value using the same process as the one used to decrease the simulated annealing temperature. Over all our experiments the combination of an adaptive diversity management with $I^P = 50$ and $N = 40$ showed the best results for 25,000 pALNS iterations, and $I^P = 100$ and $N = 10$ for 5000 pALNS iterations. The remaining parameters were directly derived from the work by Pisinger and Ropke [23].

Table 20.1 Detailed parameter setting for the pALNS algorithm for 25,000 iterations, values in parenthesis indicate adjusted values for 5000 iterations

Parameter	Value	Description
K	8	Number of threads
I^P	50 (100)	Number of parallel iterations
N	40 (10)	Maximum promising solution pool size
ϕ	0.10	Penalization for unserved customers
ξ_{min}	0.10	Minimum proportion of customers to be removed
ξ_{max}	0.40	Maximum proportion of customers to be removed
w	0.05	Reference objective degradation
p	0.5	Initial probability of accepting a degrading solution
α	0.002	Fraction of the initial temperature to be reached at the end
ρ	0.40	Reaction factor
σ_1	1.00	Score for new best solution
σ_2	0.25	Score for improving solution
σ_3	0.40	Score for non-improving accepted solution
σ_4	0.00	Score for rejected solution
l	100	Operator probability (w_θ) update frequency
$\theta_c, \theta_t, \theta_s$	1.0	A-priori relatedness weights

¹ CPU: Intel i7 860 (4 × 2.8 GHz), RAM: 6 GB DDR3, OS: Ubuntu 11.10 ×64, Java 7.

20.5.2 Validation on the D-VRPTW

To assess the effect of parallelization we tested our algorithm on the static instances for the VRPTW proposed by Solomon [29]. In the VRPTW, customers require a specific quantity of product to be delivered or picked up. The objective is to find a set of routes minimizing the traveled distance while ensuring that all customers are visited exactly once, and satisfying the capacity of the vehicle. The instances contain 100 customers located randomly (R), in clusters (C), or combining both (RC); while the planning horizon is either short (type 1) or long (type 2). These instances are organized combining location and horizon length, leading to six groups (R1, C1, RC1, R2, C2, RC2). We consider the minimization of the total distance.

Table 20.2 Comparison of gap to the best known solutions and running times for different levels of parallelization

	Seq.	Parallel: num. of threads			
		1	2	4	8
Gap	0.74%	0.72%	0.55%	0.54%	0.48%
Gap (st. dev.)	0.87%	0.88%	0.76%	0.70%	0.66%
Time (s)	36.58	37.32	22.07	14.70	11.32
Time (s, st. dev.)	6.27	6.33	4.06	2.72	2.15

Table 20.2 presents aggregated values over the 56 instances, with ten run per instance and 25,000 ALNS iterations.² The column labeled “Seq.” corresponds to the original sequential implementation of the ALNS, and the columns labeled “1” through “8” to the parallel implementation with 1–8 threads. The first and second rows contain the mean and standard deviation of the gap value relative to either the optimal or the best known solution. Finally, the third and fourth rows show the mean and standard deviation of the CPU times. Note that increasing the number of threads has a limited impact on the gap to the best known solutions, which is consistently around 0.6%, but it allows a reduction of running times by a factor of 3.3.

Figure 20.4 presents the box plot of the distribution of the gap and CPU times for the sequential (S) and parallel implementations with 1, 2, 4, and 8 threads. A graphical analysis shows that the median and variance of the gap slightly decrease with the number of threads. In contrast, the median and variance of the running time sharply decrease with the number of threads. Therefore, we selected the configuration with 8 threads as it offers the best compromise between speed and quality.³ These results show that pALNS consistently produces results that are very close to the optimal solutions of the Solomon [29] instances within a tight time budget.

² To ensure that $I = I^m \times I^p \times K \simeq 25,000$, we used $I^m = \left\lceil \frac{25,000}{40 \times K} \right\rceil$.

³ Note that the processor used is a quad-core with Intel hyper-threading technology which allows two threads per core. This partially explains the relatively small reduction of CPU times when switching from 4 to 8 threads.

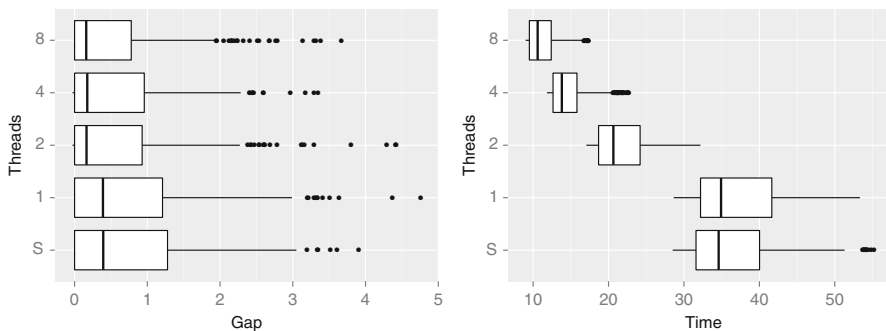


Fig. 20.4 Impact of the number of threads on the gap and CPU time

We tested the RpALNS approach on the instances proposed by Lackner [18] and based on the Solomon [29] benchmark, in which a fraction of the customers is revealed dynamically. This fraction, also referred to as degree of dynamism (δ) is either 10%, 30%, 50%, 70%, or 90%. For each instance, we performed 10 simulations in which pALNS is first run for 25,000 iterations to produce an initial solution. Then, each time a new customer appears, pALNS runs for 5000 iterations to produce a solution that will be used until the next customer is revealed. Finally, pALNS is run for 50,000 iterations to solve the a-posteriori problem, in which all the accepted customers are assumed to be known beforehand.

Table 20.3 presents the *Value of Information* (VI) [19] for each instance group and degree of dynamism (δ). The value of information for instance I is defined as the ratio $\frac{z(I) - z(I_{ap})}{z(I_{ap})}$, where $z(I)$ is the value of the solution found by the algorithm for the dynamic instance and $z(I_{ap})$ is the value of the solution for the a-posteriori instance I_{ap} . As expected, results indicate that the VI increases with the degree of dynamism, which can be explained by the fact that suboptimal routing decisions add up over time, and more decisions are made in highly dynamic instances. However, it is of only 2.1% when 10 out of 100 customers appear dynamically ($\delta = 10\%$), 4.5% for $\delta = 30\%$, and even with $\delta = 90\%$, the VI is of just 11% on average, which means that the algorithm produces a final routing that is very close to what would have been done if all the customers were known from the beginning of the day. The results also indicate the robustness of the approach with respect to the different types of instances. With the exception of group C2, for which the results are slightly better, the approach has a similar performance on all groups for a given degree of dynamism.

Table 20.4 presents a comparison of approaches for the Lackner [18] instances. The third and fourth columns present the total distance (Dist.) and number of rejected customers (Rej.) for RpALNS, averaged over 10 runs and for each group and degree of dynamism. The fifth and sixth columns report the average distance, relative average additional distance (in parenthesis), and number of rejected customers for the Large Neighborhood Search (LNS) approach proposed by Hong [15], while the seventh and eighth columns report the same values for the Genetic Algorithm

Table 20.3 Average value of information for the Lackner [18] instances

δ	R1 (%)	C1 (%)	RC1 (%)	R2 (%)	C2 (%)	RC2 (%)	Avg. (%)
10	2.05	2.89	3.06	1.70	1.66	1.61	2.14
30	4.67	5.83	5.83	4.34	1.74	4.70	4.54
50	6.41	9.28	9.03	8.15	2.82	5.38	6.93
70	8.29	11.18	10.24	10.17	5.41	8.60	9.03
90	9.33	12.49	11.84	11.83	6.51	12.33	10.71

(GA) developed by Lackner [18].⁴ Figures show that our approach is competitive both in terms of total distance and number of rejected customers. The improvement on the total distance is on average of 6.75% and 15.61% in comparison with Hong [15] and Lackner [18] respectively, and ranges from 0.56% to 34.45% depending on the instance group. This significant range of improvements combined with the previous conclusions suggest that RpALNS is in addition more robust than the other two approaches. In addition, average running times are of just 5.3 s for the initial optimization, and 2.0 s for subsequent reoptimizations, which is significantly less than the 33 and 47 s reported by Hong [15]⁵ and Lackner [18],⁶ respectively.

20.5.3 Results on the D-TRSP

We tested the RpALNS approach on a set of 280 randomly generated instances based on the Solomon [29] testbed. The 56 static instances were originally presented by the authors in Pillac et al. [22] and contain 100 requests located randomly (R), in clusters (C), or combining both (RC); while the planning horizon is either short (type 1) or long (type 2). These instances are organized combining location and horizon (i.e., C1, C2, R1, R2, RC1, and RC2). We considered 5 skills, 5 tools, and 5 types of spare parts. For each request, we selected 1 skill, and between 0 and 2 tools and spare part types. Each one of the 25 technicians has between 2 and 4 skills, and an initial set of 0–5 tools, and 2–5 spare parts. In addition, we generated release dates for either 10, 30, 50, 70, or 90 requests, leading to a complete testbed of 280 dynamic instances.

We compare the proposed approach with a regret-3 heuristic, which is used to model the solution that a human dispatcher could produce. This simple approach starts with the same initial solution as RpALNS. Each time a new request appears, it attempts to insert it in the current solution using a regret heuristic, rejecting it if it cannot be inserted. The parameter setting for RpALNS is the same as in the D-VRPTW experiments.

⁴ Note that the experimental setting of the two cited studies is not explicitly presented, which limits the relevance of direct comparisons.

⁵ CPU: Intel Core 2 Duo CPU (2.40 GHz), RAM: 2 GB, OS: Windows Vista, C#.

⁶ CPU: Pentium 4 (2.8 GHz), RAM: 1 GB, OS: Unspecified.

Table 20.4 Comparison of approaches for the Lackner [18] instances

Group	δ	RpALNS		Hong [15]		Lackner [18]	
		Dist.	Rej.	Dist.	Rej.	Dist.	Rej.
R1	10	1197.4	0.25	1257.1 (4.99%)	0.17	1278.1 (6.74%)	0.47
	30	1212.9	0.80	1286.6 (6.08%)	0.58	1337.9 (10.30%)	0.72
	50	1225.0	1.25	1295.8 (5.78%)	0.67	1330.0 (8.57%)	0.78
	70	1237.3	1.71	1331.3 (7.60%)	1.75	1336.1 (7.98%)	0.94
	90	1230.1	2.55	1335.9 (8.60%)	2.33	1278.3 (3.92%)	0.75
C1	10	850.6	0.11	895.8 (5.31%)	0.22	996.4 (17.14%)	0.00
	30	874.9	0.11	962.1 (9.97%)	0.33	1066.9 (21.95%)	0.00
	50	903.4	0.11	1001.2 (10.82%)	0.22	1236.1 (36.82%)	0.00
	70	919.1	0.11	1031.7 (12.25%)	0.22	1261.3 (37.24%)	0.00
	90	929.9	0.11	1039.8 (11.81%)	0.22	1479.6 (59.11%)	0.00
RC1	10	1389.4	0.04	1436.2 (3.37%)	1.13	1426.9 (2.70%)	0.46
	30	1421.5	0.28	1492.2 (4.98%)	1.13	1439.7 (1.28%)	0.42
	50	1463.4	0.23	1514.7 (3.50%)	1.38	1448.1 (-1.05%)	0.46
	70	1470.1	0.58	1511.3 (2.80%)	1.88	1488.4 (1.25%)	0.58
	90	1495.5	0.51	1513.9 (1.23%)	2.00	1475.2 (-1.36%)	0.42
R2	10	893.0	0.00	950.0 (6.39%)	0.09	1052.9 (17.90%)	0.03
	30	915.6	0.00	985.5 (7.63%)	0.00	1085.4 (18.54%)	0.15
	50	948.6	0.00	1016.5 (7.17%)	0.00	1138.8 (20.05%)	0.21
	70	967.7	0.00	1032.0 (6.65%)	0.09	1116.9 (15.42%)	0.30
	90	981.7	0.00	1047.8 (6.73%)	0.09	1193.3 (21.55%)	0.52
C2	10	597.2	0.00	594.7 (-0.42%)	0.00	629.1 (5.35%)	0.00
	30	597.6	0.00	651.4 (9.01%)	0.00	632.3 (5.81%)	0.04
	50	604.0	0.00	605.0 (0.17%)	0.00	689.3 (14.12%)	0.13
	70	619.2	0.00	636.5 (2.79%)	0.00	743.8 (20.12%)	0.21
	90	625.7	0.00	636.8 (1.78%)	0.00	792.5 (26.66%)	0.29
RC2	10	1024.4	0.00	1103.3 (7.70%)	0.00	1220.9 (19.18%)	0.00
	30	1053.1	0.00	1166.0 (10.73%)	0.25	1244.9 (18.21%)	0.04
	50	1060.5	0.00	1190.5 (12.26%)	0.13	1244.9 (17.38%)	0.00
	70	1091.4	0.00	1239.5 (13.57%)	0.00	1269.3 (16.30%)	0.00
	90	1130.3	0.00	1257.2 (11.23%)	0.13	1346.8 (19.16%)	0.13
Average			0.29	(+6.75%)	0.50	(+15.61%)	0.27

The static TRSP problem [22] considers the minimization of the total working time. In a dynamic setting, this objective leads to the premature ending of routes: technicians are sent home as soon as possible to reduce the duration of their route, ignoring the fact that additional requests may appear in the future. To prevent this behavior we define a *cutoff policy* that ensures for an instance I that technicians will no be sent back to their home until time $t_c(I)$. Considering that each instance has a different horizon $[0, T(I)]$, we define the relative cutoff $\alpha(G)$ for instance group G . The value of $\alpha(G)$ is defined such that all requests of instance $I \in G$ will be known before $\alpha(G)T(I)$ with a certain probability. In our experiments, $\alpha(G)$ corresponds

to the 90-percentile of the distribution of $\left\{ \frac{rd_{max}^I}{T(I)} \right\}_{I \in G}$ where rd_{max}^I is the last release date for instance I .⁷

A direct consequence of this policy is that the minimal route duration for instance I is either 0 (if the technician is not used), or $\alpha(G)T(I)$. Therefore the total duration at the end of the day is significantly longer than the one found when solving the static problem.

Table 20.5 Average gap to a-posteriori solution and number of rejected requests for the D-TRSP instances minimizing the total duration

δ	RpALNS		Regret-3	
	Gap (%)	Rej.	Gap (%)	Rej.
10%	65.7	0.0	59.9	0.4
30%	79.5	0.1	84.6	0.6
50%	93.0	0.1	100.4	1.0
70%	100.3	0.2	113.8	1.4
90%	102.8	0.4	122.3	1.8
Avg.	88.3	0.2	96.2	1.0

Table 20.5 reports the results for the 56 instances and 5 degrees of dynamism when minimizing the total duration. The first column contains the degree of dynamism (δ). The second and third columns report the average gap to an a-posteriori solution⁸ and the average number of rejected requests (Rej.) for the RpALNS. The fourth and fifth columns contain these statistics for the regret-3 heuristic. Note that running times for RpALNS are of 12 s on average for the calculation of the initial solution and 2.8 s for subsequent reoptimizations, while decision times are negligible for regret-3.

Firstly, it can be observed that gaps are large regardless of the approach. This is due to the fact that the a-posteriori solution does not consider the cutoff strategy enforced in the dynamic context. Therefore the gap should not be interpreted as an absolute performance metric, but instead as a metric that allows comparisons between approaches. Secondly, the results show that, as expected, both the gap and number of rejected requests increase with the degree of dynamism. Finally, they indicate that the RpALNS approach leads to better solutions both in terms of quality of the routing (measured by the gap) and ability to cope with new requests (measured by the number of rejected requests).

The cutoff policy forces technicians to wait at their current location before returning home. Thus, the minimization of the working time may not be as relevant

⁷ With this definition: $\alpha^{C1} = 0.380$, $\alpha^{C2} = 0.509$, $\alpha^{R1} = 0.357$, $\alpha^{R2} = 0.419$, $\alpha^{RC1} = 0.321$, $\alpha^{RC2} = 0.400$.

⁸ The gap for instance I is defined as the ratio $\frac{z(I) - \underline{z}(I)}{\underline{z}(I)}$ where $z(I)$ is the value of the solution found by the algorithm for the dynamic instance, and $\underline{z}(I)$ is the solution to the static a-posteriori instance with 50,000 iterations of the pALNS algorithm.

in a dynamic context as it is for the static case. To assess the validity of this objective, we performed the same experiments with the minimization of the total distance as unique objective.

Table 20.6 Average gap to a-posteriori solution and number of rejected requests for the D-TRSP instances minimizing the total distance

δ	RpALNS		Regret-3	
	Gap (%)	Rej.	Gap (%)	Rej.
10	2.4	0.1	10.5	0.3
30	5.4	0.1	30.5	0.4
50	10.8	0.3	44.1	1.0
70	11.8	0.2	57.5	1.2
90	17.9	0.4	64.1	1.4
Avg.	9.7	0.2	41.3	0.8

Table 20.6 compares the two approaches when the objective only considers the minimization of the total distance. As before, the gap and number of rejected requests generally increases with the degree of dynamism. These results show that RpALNS consistently outperforms the regret-3 heuristic.

Table 20.7 Difference in total working time and distance when minimizing the total distance instead of the total working time (in %)

δ	RpALNS		Regret-3	
	Δ_{WT}	Δ_{Dist}	Δ_{WT}	Δ_{Dist}
10	-8.0	-40.9	-1.4	-33.6
30	-9.8	-45.5	-7.7	-31.7
50	-16.4	-46.5	-11.0	-31.4
70	-18.5	-47.6	-10.8	-30.2
90	-20.2	-45.4	-11.9	-32.0
Avg.	-14.6	-45.2	-8.5	-31.8

Finally, Table 20.7 presents the effect of minimizing the total distance instead of the working time on total working time (Δ_{WT}) and total distance (Δ_{Dist}) for the two approaches. As expected, minimizing the distance instead of the working time leads to a reduction of the total distance by 45% and 32% for RpALNS and regret-3, respectively. More surprisingly, it also leads to a reduction of the total working time by 15% and 8%. This can be explained by the cutoff policy that is contradictory with the minimization of the working time, which mainly focuses on minimizing waiting times. In contrast, minimizing the total distance always leads to a reduction of the travel times, which in turn reduces the duration of routes.

20.6 Conclusions

In this paper we introduced a new dynamic optimization problem, namely the Dynamic Technician Routing and Scheduling Problem (D-TRSP). This problem arises in several practical contexts such as public utilities, telecommunications, and maintenance operations.

We proposed a periodic reoptimization approach based on a parallel Adaptive Large Neighborhood Search (RpALNS) that produces a new routing plan each time a new request appears. Our preliminary computational results indicate that RpALNS dominates a simpler regret-3 heuristic, by yielding high quality results in limited time. In addition, its relative simplicity makes it a good candidate for practical applications.

We illustrated that the minimization of the total working time, although perfectly sound in a static context, does not fit well in a dynamic environment. In particular, we have shown that minimizing the total distance ultimately leads to solutions that are better both in terms of total distance and duration.

Further work will focus on developing approaches that continuously optimize the routing throughout the day. In addition, we are testing the proposed approach on real world data from an industrial partner. Finally, uncertainty should be modeled to better anticipate the arrival of new requests and improve the quality of the decisions.

Acknowledgements Financial support for this work was provided by the CPER Vallée du Libre (Contrat de Projet Etat Region, France); and the CEIBA (Centro de Estudios Interdisciplinarios Básicos y Aplicados en Complejidad, Colombia). This support is gratefully acknowledged. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. A. Attanasio, J.F. Cordeau, G. Ghiani, G. Laporte, Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Comput.* **30**(3), 377–387 (2004). doi:10.1016/j.parco.2003.12.001.
2. J. Barcelo, H. Grzybowska, S. Pardo, Vehicle routing and scheduling models, simulation and city logistics, in *Dynamic Fleet Management*, ed. by V. Zempekis, C.D. Tarantilis, G.M. Giaglis, I. Minis. *Operations Research/Computer Science Interfaces*, vol. 38 (Springer, New York, 2007), pp. 163–195
3. A. Beaudry, G. Laporte, T. Melo, S. Nickel, Dynamic transportation of patients in hospitals. *OR Spectr.* **32**, 77–107 (2010). doi:10.1007/s00291-008-0135-6
4. R. Bent, P. Van Hentenryck, Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Oper. Res.* **52**(6), 977–987 (2004)
5. I. Benyahia, J.Y. Potvin, Decision support for vehicle dispatching using genetic programming. *IEEE Trans. Syst. Man Cybern. A: Syst. Humans* **28**(3), 306–314 (1998)
6. F. Blakeley, B. Arguello, B. Cao, W. Hall, J. Knolmayer, Optimizing periodic maintenance operations for Schindler elevator corporation. *Interfaces* **33**(1), 67–79 (2003). doi:10.1287/inte.33.1.67.12722

7. Y. Borenstein, N. Shah, E. Tsang, R. Dorne, A. Alsheddy, C. Voudouris, On the partitioning of dynamic workforce scheduling problems. *J. Sched.* **13**(4), 411–425 (2010). doi:10.1007/s10951-009-0152-6
8. N. Bostel, P. Dejax, P. Guez, F. Tricoire, Multiperiod planning and routing on a rolling horizon for field force optimization logistics, in *The Vehicle Routing Problem: Latest Advances and New Challenges*, ed. by R. Sharda, B. Golden, S. Raghavan, E. Wasil. Operations Research/Computer Science Interfaces, vol. 43 (Springer, New York, 2008), pp. 503–525
9. M.S. Chang, S. Chen, C. Hsueh, Real-time vehicle routing problem with time windows and simultaneous delivery/pickup demands. *J. East. Asia Soc. Transp. Stud.* **5**, 2273–2286 (2003)
10. B.K.S. Cheung, K.L. Choy, C.-L. Li, W. Shi, J. Tang, Dynamic routing model and solution methods for fleet management with mobile technologies. *Int. J. Prod. Econ.* **113**(2), 694–705 (2008). doi:10.1016/j.ijpe.2007.10.018
11. J.-F. Cordeau, G. Laporte, F. Pasin, S. Ropke, Scheduling technicians and tasks in a telecommunications company. *J. Sched.* **13**(4), 393–409 (2010). doi:10.1007/s10951-010-0188-7
12. M. Gendreau, F. Guertin, J.-Y. Potvin, E. Taillard, Parallel tabu search for real-time vehicle routing and dispatching. *Transp. Sci.* **33**(4), 381–390 (1999). doi:10.1287/trsc.33.4.381
13. A. Haghani, S. Jung, A dynamic vehicle routing problem with time-dependent travel times. *Comput. Oper. Res.* **32**(11), 2959–2986 (2005). doi:10.1016/j.cor.2004.04.013
14. H. Hashimoto, S. Boussier, M. Vasquez, C. Wilbaut, A GRASP-based approach for technicians and interventions scheduling for telecommunications. *Ann. Oper. Res.* **183**, 143–161 (2011). doi:10.1007/s10479-009-0545-0
15. L. Hong, An improved LNS algorithm for real-time vehicle routing problem with time windows. *Comput. Oper. Res.* **39**(2), 151–163 (2012). doi:10.1016/j.cor.2011.03.006
16. S. Ichoua, M. Gendreau, J.-Y. Potvin, Diversion issues in real-time vehicle dispatching. *Transp. Sci.* **34**(4), 426–438 (2000). doi:10.1287/trsc.34.4.426.12325
17. S. Ichoua, M. Gendreau, J.-Y. Potvin, Vehicle dispatching with time-dependent travel times. *Eur. J. Oper. Res.* **144**(2), 379–396 (2003). doi:10.1016/S0377-2217(02)00147-9
18. A. Lackner, Dynamische Tourenplanung mit ausgewählten Metaheuristiken. PhD thesis, Georg-August-Universität Göttingen, 2004
19. K. Lund, O.B.G. Madsen, J.M. Rygaard, Vehicle routing problems with varying degrees of dynamism. Technical report, IMM Institute of Mathematical Modelling (1996)
20. R. Montemanni, L.M. Gambardella, A.E. Rizzoli, A.V. Donati, Ant colony system for a dynamic vehicle routing problem. *J. Comb. Optim.* **10**(4), 327–343 (2005). doi:10.1007/s10878-005-4922-6
21. V. Pillac, M. Gendreau, C. Guéret, A.L. Medaglia, A review of dynamic vehicle routing problems. *Eur. J. Oper. Res.* **225**(1), 1–11 (2013). doi:10.1016/j.ejor.2012.08.015
22. V. Pillac, C. Guéret, A.L. Medaglia, A parallel matheuristic for the technician routing and scheduling problem. *Optim. Lett.* **7**(7), 1525–1535 (2013). doi:10.1007/s11590-012-0567-4
23. D. Pisinger, S. Ropke, A general heuristic for vehicle routing problems. *Comput. Oper. Res.* **34**(8), 2403–2435 (2007). doi:10.1016/j.cor.2005.09.012
24. D. Pisinger, S. Ropke, Large neighborhood search, in *Handbook of Metaheuristics*, ed. by M. Gendreau, J.-Y. Potvin. International Series in Operations Research and Management Science, vol. 146 (Springer, New York, 2010), pp. 399–419
25. J.-Y. Potvin, J.-M. Rousseau, A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *Eur. J. Oper. Res.* **66**(3), 331–340 (1993). doi:10.1016/0377-2217(93)90221-8
26. C. Prins, Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Eng. Appl. Artif. Intell.* **22**(6), 916–928 (2009). doi:10.1016/j.engappai.2008.10.006
27. S. Ropke, D. Pisinger, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **40**(4), 455–472 (2006)
28. P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, in *Principles and Practice of Constraint Programming – CP98*. Lecture Notes in Computer Science, vol. 1520 (Springer, Berlin/Heidelberg, 1998), pp. 417–431

29. M.M. Solomon, Algorithms for the vehicle-routing and scheduling problems with time window constraints. *Oper. Res.* **35**(2), 254–265 (1987)
30. E.D. Taillard, L.M. Gambardella, M. Gendreau, J.-Y. Potvin, Adaptive memory programming: a unified view of metaheuristics. *Eur. J. Oper. Res.* **135**(1), 1–16 (2001). doi:10.1016/S0377-2217(00)00268-X
31. H. Tang, E. Miller-Hooks, R. Tomastik, Scheduling technicians for planned maintenance of geographically distributed equipment. *Transp. Res. E: Logist. Transp. Rev.* **43**(5), 591–609 (2007). doi:10.1016/j.tre.2006.03.004
32. F. Tricoire, Optimisation des Tournées de Véhicules et de Personnels de Maintenance: Application à la Distribution et au Traitement des Eaux. PhD thesis, École Nationale Supérieure des Techniques Industrielles et des Mines de Nantes, 2006
33. E. Tsang, C. Voudouris, Fast local search and guided local search and their application to British Telecom's workforce scheduling problem. *Oper. Res. Lett.* **20**(3), 119–127 (1997). doi:10.1016/S0167-6377(96)00042-9
34. J.I. Van Hemert, J.L. Poutré, Dynamic routing problems with fruitful regions: models and evolutionary computation, in *Parallel Problem Solving from Nature*, ed. by X. Yao, E. Burke, J.A. Lozano, J. Smith, J.J. Merelo-Guervós, J.A. Bullinaria, J. Rowe, P. Tino, A. Kabán, H.-P. Schwefel. Lecture Notes in Computer Science, vol. 3242 (Springer, Berlin/Heidelberg, 2004), pp. 692–701
35. T. Vidal, T.G. Crainic, M. Gendreau, C. Prins, A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Comput. Oper. Res.* **40**(1), 475–489 (2013). doi:10.1016/j.cor.2012.07.018
36. D. Weigel, B. Cao, Applying GIS and OR techniques to solve Sears technician-dispatching and home delivery problems. *Interfaces* **29**(1), 112–130 (1999). doi:10.1287/inte.29.1.112
37. J. Xu, S. Chiu, Effective heuristic procedures for a field technician scheduling problem. *J. Heuristics* **7**(5), 495–509 (2001)

Chapter 21

Optimized Air Routes Connections for Real Hub Schedule Using SMPSO Algorithm

H. Rahil, B. Abou El Majd, and M. Bouchoum

Abstract The choice to open new routes for air carriers, airports and regional governments have some tools to promote desirable connections to be offered toward specific destinations. With a given flight program, the air carrier decision to open new routes faces several constraints and affects the flight schedules in a remarkable way. This work is distinguished by the fact of being the first to introduce the problem of connectivity in the network of an airline whose main activity is based on air hub structure, optimizing the insertion of new airline routes while ensuring the best fill rate seats and avoiding significant delays during correspondence. Quality of Service Index (QSI) will be considered as a dual parameter for the profit of a new opened market. This aspect of decision making is formulated as multi-objective problem by testing the impact of a new insertion in term of delays, generated with related costs and financial gain, and the quality of service offered to a target customers. The SMPSO Algorithm is adopted to generate a Pareto-optimal front composed of many optimal departure times toward the new opening insuring the best filling ratio with minimum connecting times. Experiences are based on real instance of Royal Air Maroc flights schedule on the hub of Casablanca.

Keywords Route networks • Hub and spokes • Outbound/Inbound connection • Flight schedule • Multi-objective optimization • SMPSO algorithm • Pareto optimal

21.1 Introduction

In literature on air transport there is not a precise definition of connectivity. Typically, connectivity measures allow to identify how it is easy to reach the rest of the network starting from an airport or which are the opportunity for interconnections that the

H. Rahil (✉) • B. Abou El Majd • M. Bouchoum
Laboratory of Computer Science and Decision Support, Faculty of Sciences, Casablanca, Morocco
e-mail: hichamrahil@gmail.com; b.abouelmajd@fsac.ac.ma;
m.bouchoum@fsac.ac.ma

airport offers with the latter typically employed in order to measure performance of airline hub [1, 6, 10]. Connectivity is important for airport, airlines and local authorities. For airports, connectivity is employed to benchmark their performance against other airports [8] to better understand on which new opened routes they can provide a competitive hub service and to evaluate self help strategies. More in general, as discussed by Burghouwt [2] connectivity has an important role in setting strategic airport planning.

Connectivity measure can be applied to a single airlines network thus allowing a performance comparison among airlines, a rivalry analysis and a better understanding of benefits coming from network consolidation as a result of alliances and partnerships [17]. For policy makers, connectivity measures allow to monitor the level of service provided to the local area, to evaluate the cohesion to the rest of the country for example in term of travel times required to reach a given share of country's gross domestic product (GDP) [13]. Burghouwt and Redondi analysis [3] showed that connectivity measures differ from traditional size based measures typically employed to rank airport and to proxy their competitive position. Connectivity thus needs specific indexes like the Airport Connectivity Quality Index (ACQI) introduced by Wittman and Swelbar [18]. The connectivity could be also represented according to the graph theory as the number of step required to reach a destination. The standard connectivity index can be weighted by service frequency or by the number of seat offered. In order to develop measure that more realistically represent the connectivity chance for passengers, several adjustments can be imposed in order to account for temporal coordination and connection quality. Thresholds can be imposed in term of maximum number of steps and minimum and maximum connecting time at intermediate airports. For its part, Royal Air Maroc, the Morocco's flag carrier is weighing whether global alliance membership will help attract more premium travellers and provide connectivity to other parts of the world. In addition to its flights to over 80 destinations, Royal Air Maroc now offers more flights around the world through many airline partners. An alliance would benefit from RAM's position in Morocco but also central/west Africa, RAM's largest international market after Europe as illustrated in Fig. 21.1.

Adding connections from Casablanca to elsewhere in Africa would likely provide only incremental traffic. With most of RAM's markets served less than daily as illustrated in Fig. 21.2, it will be challenged to offer convenient connections. Even on only the Beijing-Casablanca leg, RAM will compete with many low-priced one-stop services through Europe and the Gulf. To be both competitive and safe, airlines must be managed with just. To do this, they must use specific optimization techniques at each stage of production.

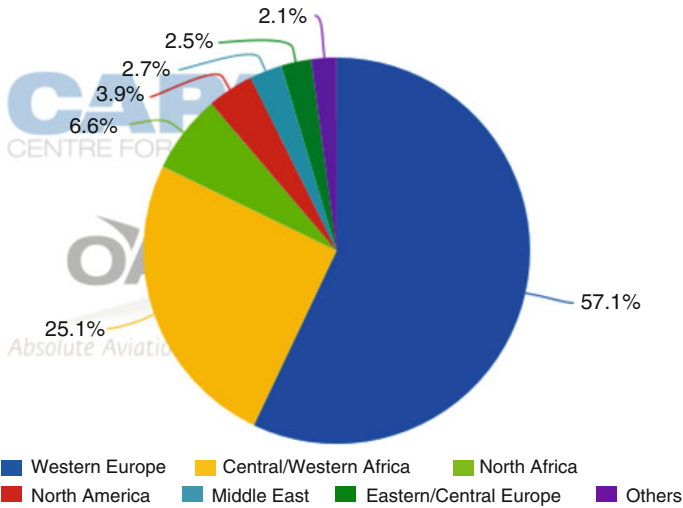


Fig. 21.1 Morocco international seat capacity share by alliance: 5-Jan-2015 to 11-Jan-2015

These mathematical techniques are called “operational research”. This area was born under the influence of Anglo-Saxon military needs during the Second World War, with the beginnings of computers and methods known as linear programming. Operational research has since developed a lot and has largely penetrated the world of business and industry. Given the stakes, methods are sometimes confidential. The standard connectivity index can be weighted by service frequency or by the number of seat offered. The connectivity process could be represented according to the graph theory as the number of step required to reach a destination.

In order to develop measures that more realistically represent the connectivity chance for passengers, several adjustments can be imposed in order to account for temporal coordination and connection quality. Thresholds can be imposed in term of maximum number of steps and minimum and maximum connecting time at intermediate airports. For an airline, the flight schedule, specifying the flight steps and departure time of each flight segment, defined broadly competitive position, it is therefore a determining factor in the profitability of airlines. Most airlines use an airport as a main base of operations. This is the place where are the technical facilities maintenance of aircraft and often their sales office. These main bases are also, of course, interchanges but if many companies have adopted the term hub there is relatively little that ensure speed and guaranteed connections [5, 7]. The transfer between flights is the passenger’s responsibility and in case of delay the second segment is lost [11]. Passengers obviously prefer direct flights, that’s why The minimum connecting time is a very important parameter that defines a competitive appearance in the sale of tickets between air carriers and predicts the connectivity gain of each new route. Several editions of this work have been presented at international and national conferences [14, 15] and during these scientific meetings, it was

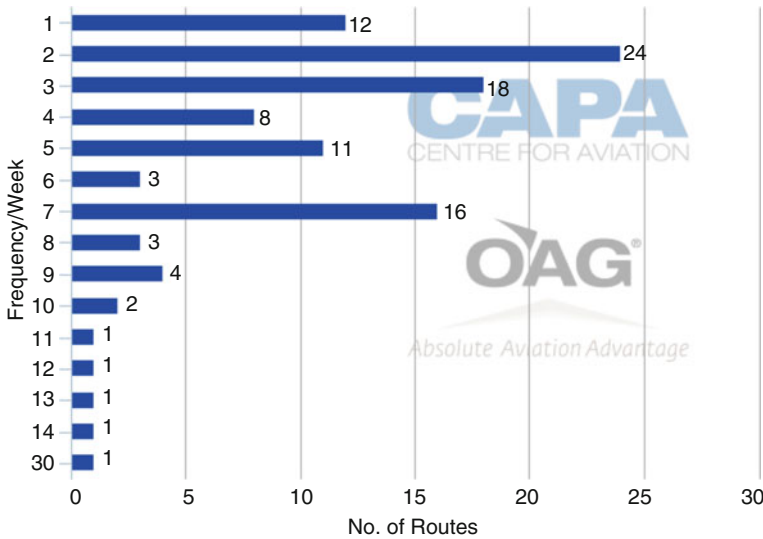


Fig. 21.2 Royal Air Maroc routes served by frequency/week: 12-Jan-2015 to 18-Jan-2015

mentioned that this work is distinguished by the fact of being the first to introduce the problem of connectivity in the network of an airline whose main activity is based on air hub structure, optimizing the insertion of new airline routes while ensuring the best fill rate seats and avoiding significant delays during correspondences. Quality of Service Index (QSI) will be considered as a dual parameter for the profit of a new opened market.

This aspect of decision making is formulated as multi-objective problem by testing the impact of a new insertion in term of delays, generated with related costs and financial gain, and the quality of service offered to a target customers. The chapter is organized as follows: we start with an exhaustive introduction of flight schedule management then we present in Sect. 21.2 a mathematical formulation for adding new routes to an existent programm, under specific constraints. Section 21.2.1 presents the optimization process used to solve the problem, while Sect. 21.3 gives some experimental results that further explain the model and the optimisation process, discusses the possible extensions of the model and sketches further directions of research. Finally, Sect. 21.4 concludes the paper and states some perspectives.

21.2 Methods

21.2.1 Problem Formulation

Let A_k (resp. D_k) the set of the weekly arrivals time (resp. departures time) from an airport $k \in K$ to the hub and a_{ki} the i th one (resp. d_{kj} the j th one) with

$i, j \in \{1, 2, \dots, N_k\}$ (a_{ki} and d_{kj} are classed in ascending order), and N_k is the weekly frequency of routes between hub and a given airport $k \in K$, we take the minute as unit of time. We note that, arrivals and departures time during the week are included in the interval $[0, 10080]$. Figures 21.3 and 21.4 shows respectively the arrival and departure programs of twenty airports for a typical week, in this example, the hub receives regularly and every day, one flight from every airport figuring on the targeted potentials list.

$$A_{ki} = \begin{cases} 1 & \text{Outbound connection is realized for } a_{ki} \\ 0 & \text{otherwise,} \end{cases}$$

$$D_{kj} = \begin{cases} 1 & \text{Inbound connection is realized for } d_{kj} \\ 0 & \text{otherwise,} \end{cases}$$

w_k is the weight vectors given by:

$$w_k = \frac{P_k}{\sum_{i=1}^{N_k} P_i},$$

This parameter is introduced to encourage the connection of airports with high potentials, and $x \in [x + t + t_{min}, 10080 - t_{max}]$, where t_{min} is the minimum connecting time, and t_{max} is the maximum allowed connecting time. Figure 21.5 illustrates the Outbound and Inbound connections, where the flight Inbound step (red color), pass through the hub to join the new destination, and return back again via the hub for the Outbound step (black color). During the optimization process, each random value x generates two intervals $[x - t_{max}, x - t_{min}]$ and $[x + t + t_{min}, x + t + t_{max}]$. An Outbound connection is realized when a given value a_{ki} comes inside the arrival time window $[x - t_{max}, x - t_{min}]$ and the Inbound one is realized when a given value d_{kj} comes inside departure time window $[x + t + t_{min}, x + t + t_{max}]$. Figure 21.6 gives the time intervals where the arrival and departure times should occur to guarantee a full connection.

21.2.1.1 The Best Filling Ratio Profit Objective

Network planning is an integral part of every air carrier’s revenue generation capabilities. Considering today’s challenging business environment, one way of knowing whether an airliner’s network planning and scheduling is paying off, is to see how much revenue is improving. There are several aspects of scheduling that can reduce operating costs and lift overall profitability.

In addition, the air transporter must work in conjunction with several contributors, including revenue management office, to be sure that they have a same level of understanding of the features of targeted customers in a given market. When the decision maker opts to reach the best filling ratio (maximising profit) in the new

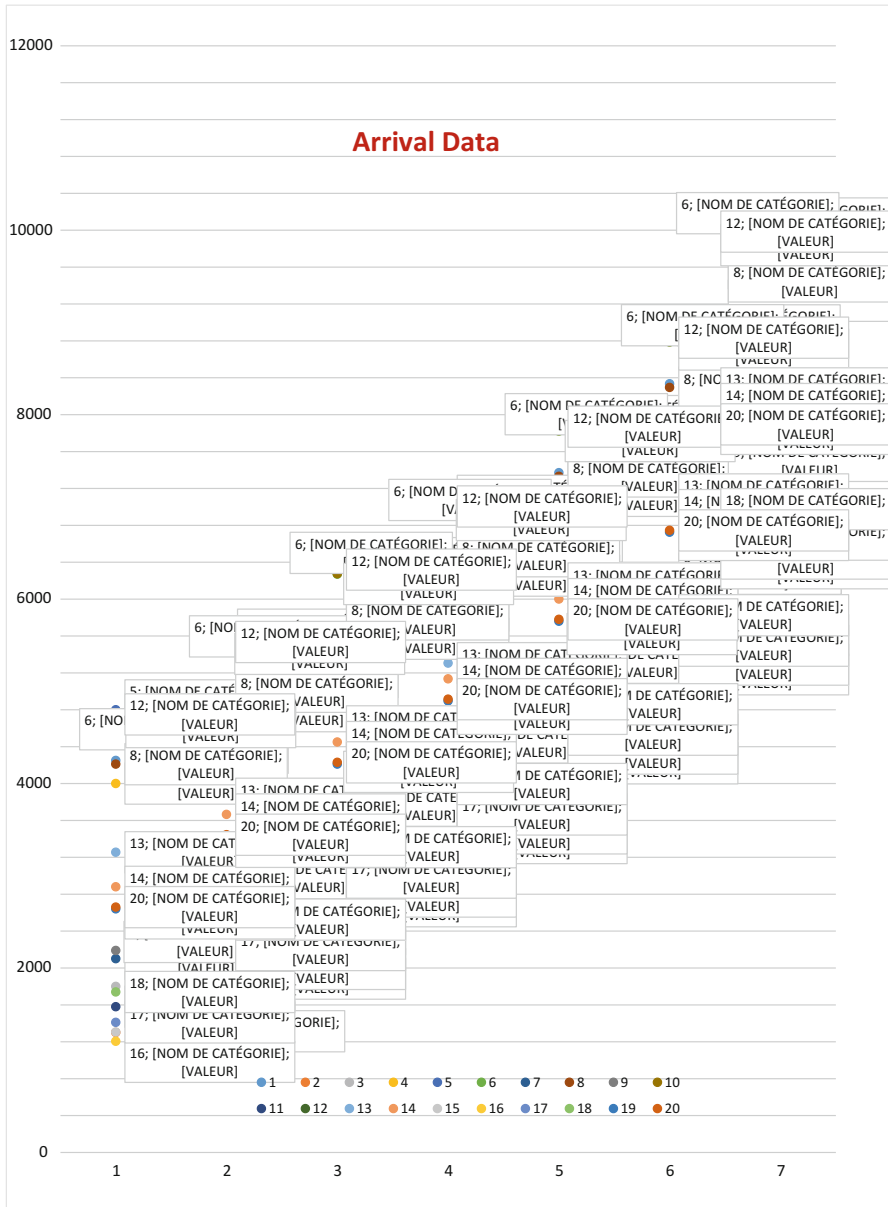


Fig. 21.3 Arrivals Distribution of twenty targeted airports over a typical week, the hub receives every day, one flight from each airport

route, he should first of all localise the category of customers interested to this new insertion and define the most important airport passenger potentials travelling to this destination. In order to find the best departure time x , we maximize the following

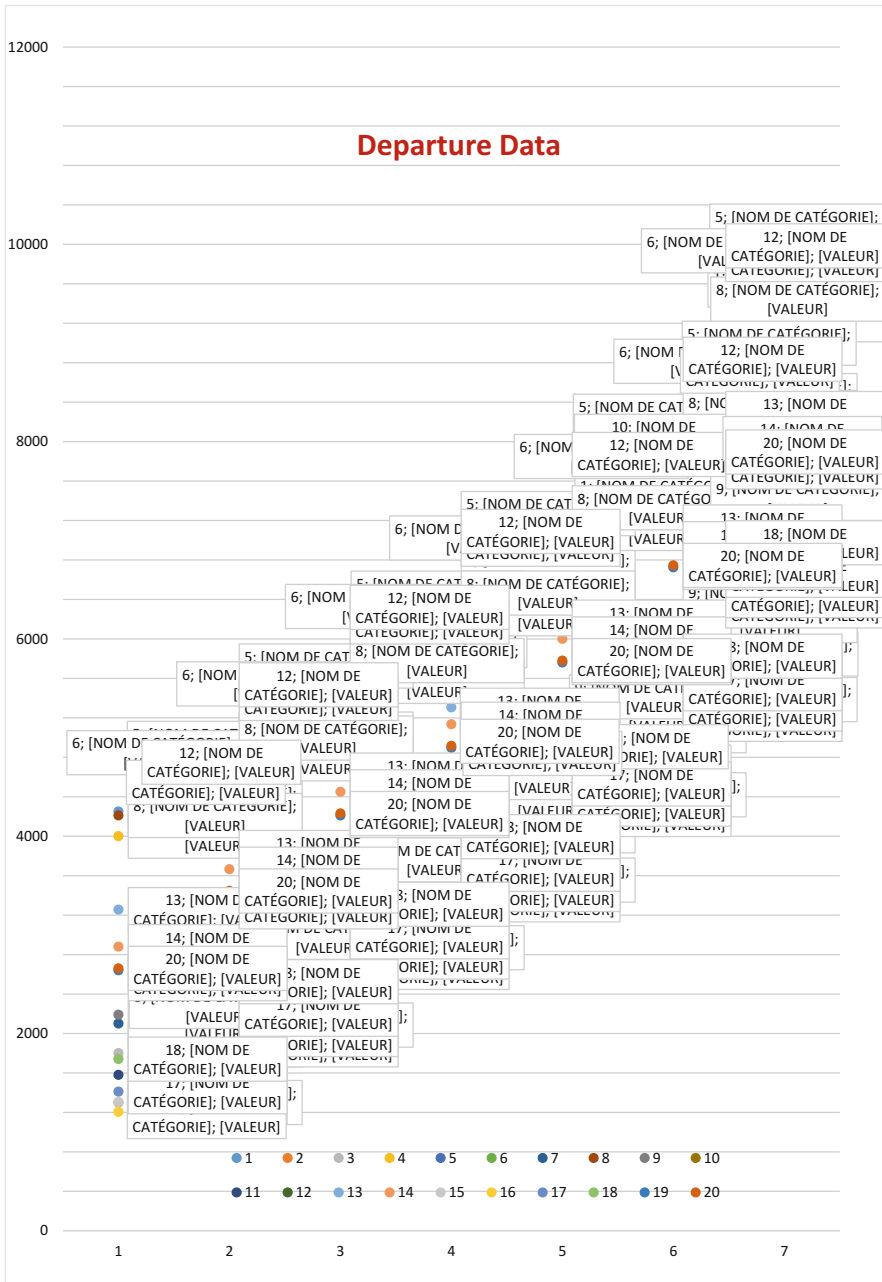


Fig. 21.4 Departures Distribution of twenty targeted airports over a typical week, the destinations are served every day from the hub

objective function

$$f_1(x) = \sum_k \sum_i \sum_j w_k A_{ki} D_{kj} \tag{21.1}$$

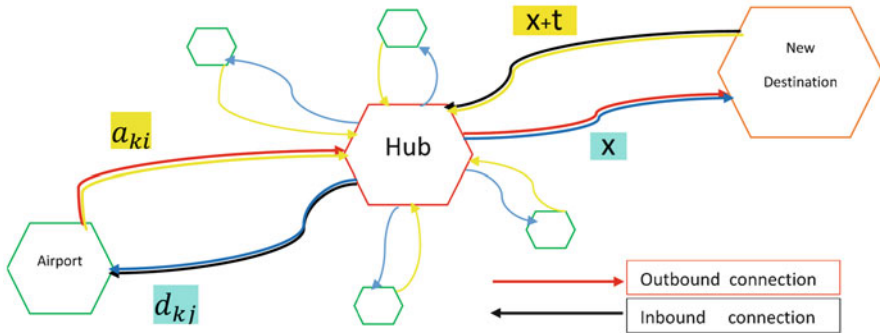


Fig. 21.5 Hub Outbound/Inbound connections: flight Inbound step (red color), pass through the hub to join the new destination, and return back again via the hub for the Outbound step (black color)

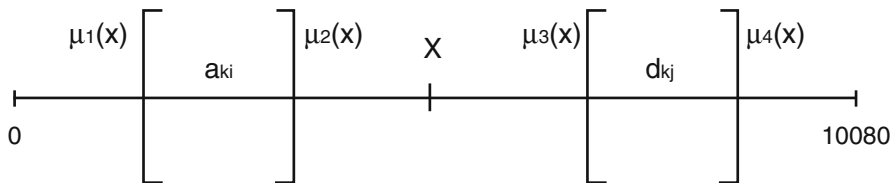


Fig. 21.6 Time windows: Time windows provides the opportunity to determine which flights can be connected

21.2.1.2 The Best Quality of Service Index Objective

Quality of service models are used to estimate the probability a traveler selects a specific itinerary connecting an airport pair. Itineraries are the products that are ultimately purchased by passengers, and hence it is the characteristics of these itineraries that influence demand. In making their itinerary choices, travelers make tradeoffs among the characteristics that define each itinerary (e.g. departure time, equipment type(s), number of stops, route, carrier, delays ...). Modeling these itinerary-level tradeoffs is essential to truly understand air-travel demand and is, therefore, one of the most important components of network-planning models. One among several aspects We take as quality of service indicator will be the generated

delays, that means, we have to decrease the delays average, caused by precedent correspondence. For a smooth connection between flights, it's important to give yourself adequate time between the arrival of the first flight and the departure of the next one.

This time between flights is known as the minimum connection time, and it's the time needed to facilitate security checks and transfers inter terminals. Connecting time is also a critical parameter for air passengers, to make there choice of convenient route correspondence, they prefer generally, the companies offering comfortable services, including minimum hub connecting times, in both outbound and inbound legs of the travel. The delays average can take the following form:

$$f_2(x) = \frac{\sum_{kij}(x - a_{ki})A_{ki} \times D_{kj}}{\sum_{kij} A_{ki} \times D_{kj}} + \frac{\sum_{kij}(d_{kj} - (x + t))A_{ki} \times D_{kj}}{\sum_{kij} A_{ki} \times D_{kj}}$$

For efficient results, the objective function calculates the average only for flights that are connected. Thus the average received will help to quantify the cumulated delays caused by these connections, and then will give us an idea about the resulting costs if we opt for this strategy.

21.2.2 SMPSO Algorithm

The term SMPSO has come to be used to refer to the metaheuristic 'Speed constrained Multi-objective PSO', and can be defined as a recent multi-objective PSO algorithm. The particle swarm optimization algorithm (PSO) is a famous population-based search algorithm which adopted in the SMPSO algorithm. It was introduced originally by R. Eberhart and J. Kennedy in 1995 [12]. The algorithm emulates the swarm behavior of bird flocking or fish schooling and belongs to the Swarm Intelligence (SI). This optimization algorithm explores the search space of a problem by moving particles, and each one of them represents a potential solution to optimize one or several objectives. The PSO has several attractive features: Simplicity, High efficiency, fast convergence, strong robustness, flexibility, easy to implement, Parallelism, etc. This advantage makes PSO algorithm more robust, and it has been successfully applied in many areas and solved a variety of optimization problems in a faster and cheaper way [16, 19]. Each particle's position represents a solution of the problem, and depends on the particle's velocity. In a search space of dimension d and at time step t , each particle i of the swarm is presented by its current position $X(t)$ given by:

$$X_i(t) = (X_i^1, X_i^2, \dots, X_i^d),$$

its best previous position $pBest$ and its fitness.

The particle moves to its own way, towards its best previous position ($pBest$) or towards the best position of all particles ($gBest$) namely the leader, by updating the velocity $V(t)$ and the position $X(t)$ using (1) and (2):

$$V_i(t+1) = w * V_i(t) + V_i^1(t+1) + V_i^2(t+1) \quad (21.2)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (21.3)$$

where,

$$V_i^1(t+1) = c_1 * r_1 * (pBest - X_i(t)),$$

$$V_i^2(t+1) = c_2 * r_2 * (gBest - X_i(t)),$$

w is the inertia weight, c_1 and c_2 are two positive constants called cognitive learning rate and social learning rate respectively, r_1 and r_2 are two uniformly distributed random values in the range $[0, 1]$. The acceleration constants c_1 and c_2 represent the weighting of the stochastic acceleration terms that pull each particle toward $pBest$ and $gBest$ positions. Therefore, each particle of the swarm is effectively moving through the search space, with the aim to obtain better results in accuracy (the quality of the resulting approximation sets) and the convergence speed. It is becoming increasingly difficult to ignore the swarm explosion when solving multi-modal problems by using a multi-objective PSO algorithm. The velocity of the particles in these problems can become too high, resulting in erratic movements towards the upper and lower limits of the positions of the particles. SMPSO used to limit the velocity of the particles by using a velocity constriction mechanism. For that, the velocity of each particle is calculated by (2), after that its multiplied by the constriction factor χ using (3) [4], then each variable j of the i th particle's velocity $V_{i,j}$ constrained by using (4).

$$\chi = \frac{2}{2 - \rho - \sqrt{\rho^2 - 4\rho}} \quad (21.4)$$

where,

$$\rho = \begin{cases} c_1 + c_2 & \text{if } c_1 + c_2 > 4 \\ 1 & \text{otherwise.} \end{cases}$$

$$V_{i,j}(t) = \begin{cases} \delta_j & \text{if } V_{i,j}(t) > \delta_j \\ -\delta_j & \text{if } V_{i,j}(t) \leq -\delta_j \\ V_{i,j}(t) & \text{otherwise.} \end{cases} \quad (21.5)$$

where,

$$\delta_j = \frac{\text{upper-limit}_j - \text{lower-limit}_j}{2}$$

As we can see in Fig. 21.7, the basic of the SMPSO algorithm is very simple. It starts by initializing randomly the swarm. According to the evaluation of each particle, it updates the leaders archive and particles memory.

As long as it has not reached the maximum iteration or some value criterion, it updates the velocity and the position of the particle by using the formulas (1), (2), (3) and (4) seen previously, and repeats until achieving one of the stopping criterion.

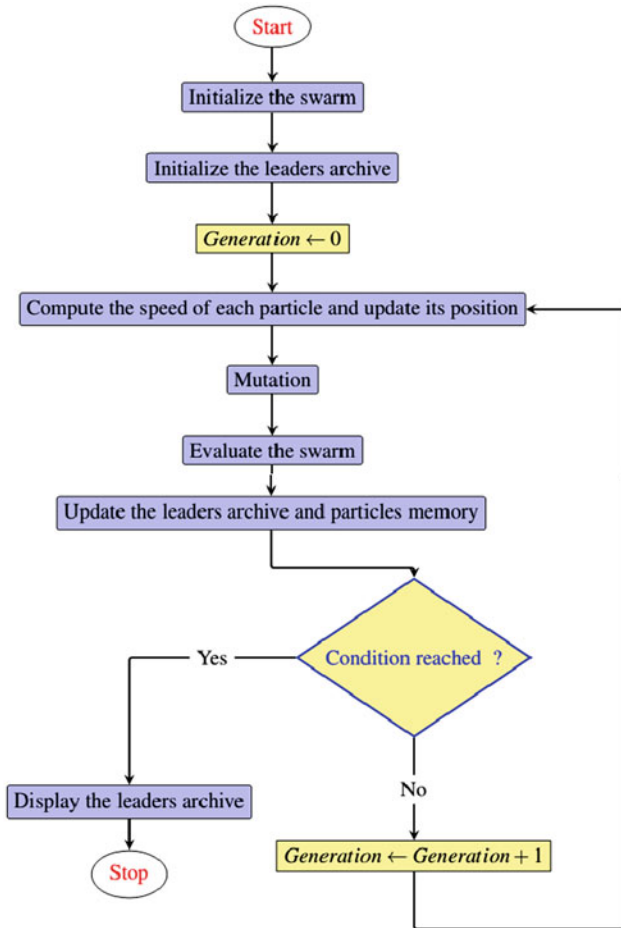


Fig. 21.7 Flowchart of SMPSO algorithm

21.3 Results and Discussion

First, we compute for different number of assessed airports. As shown in Fig. 21.8, the number of Pareto optimal solutions increases with the number of airports. This is quite normal because the chances of capturing values inside intervals $[\mu_1(x); \mu_2(x)]$ and $[\mu_3(x); \mu_4(x)]$ become greater with increasing the size of A_k and D_k . Following the air carrier demand, we focus on the test of 30 airports, obtained results are given by Fig. 21.9, moreover, Table 21.1 gives pareto's front solutions with the corresponding coordinates. We obtain seven optimal solutions, offering the possibility to connect 30 targeted airports to the new opening, maintaining an interesting fill rate and minimizing the stopover time over the hub. One of the factors sought by the potential passenger is the shortness of the trip. He therefore preferred the direct flights between the airport of departure and destination.

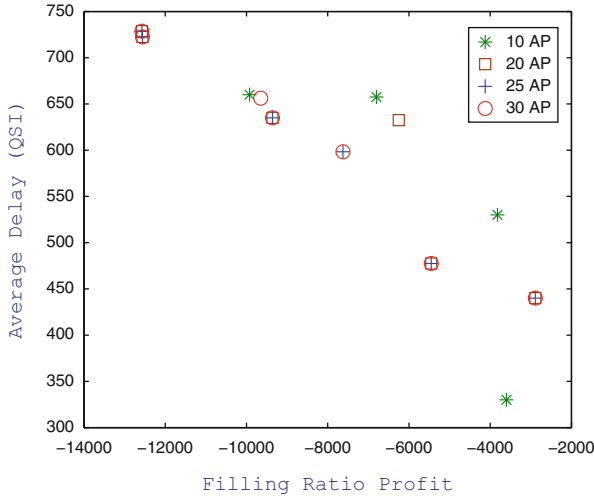


Fig. 21.8 Pareto optimal solutions, connecting 10, 20, 25 and 30 Airports (AP) , insuring the best filling ratios and reducing to minimum, the hub stopover times

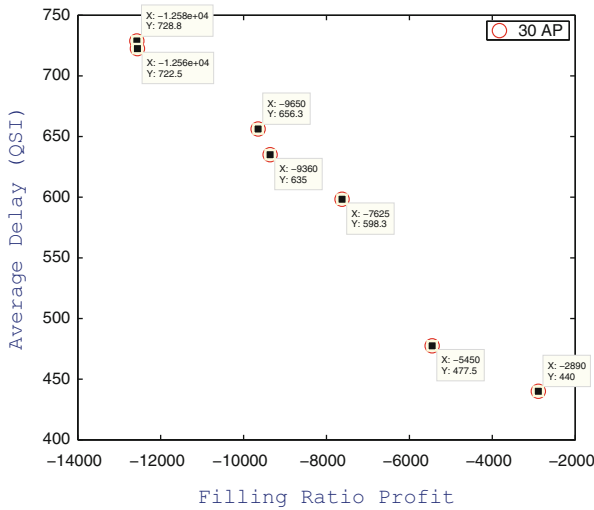


Fig. 21.9 Pareto optimal for the thirty airports case, showing the coordinates of obtained optimal solutions

When the passenger is obliged to carry out a correspondence, airlines have an incentive to attract these passengers to increase the profitability of their lines by providing a continuous service from airport of departure, passing via the connecting airport until reaching the final destination.

Tables 21.2 and 21.3 gives details of connected airports for each solution from Pareto front. We notice that Airports 5, 6, 15, 25, 26 and 28 are not connected

Table 21.1 Optimal solutions coordinates (f_1, f_2)

x	Objective function	
	f_1	f_2
$S_1 = 4710$	12,560	722.5
$S_2 = 5433$	2890	440
$S_3 = 4308$	12,575	728.75
$S_4 = 4660$	9360	635
$S_5 = 1817$	5450	477.5
$S_6 = 5156$	9650	656.25
$S_7 = 3611$	7625	598.33

Table 21.2 Optimal solutions with correspondent rotations

Optimal solutions	Flight rotation	
	Departure (UTC)	Arrival (UTC)
S_1	THU 06:30	THU 23:00
S_2	THU 18:35	FRI 11:05
S_3	WED 23:50	THU 16:20
S_4	THU 05:40	THU 22:10
S_5	TUE 06:25	TUE 22:55
S_6	THU 13:55	FRI 06:25
S_7	WED 12:15	THU 04:45

Table 21.3 Table of connected airports with correspondent optimal solutions

AP	1	2	3	4	7	8	9	10	11	12	13	14	16	17	18	19	20	21	22	23	24	27	29	30	
S_1	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓			✓			✓	✓	✓		✓	✓	
S_2				✓					✓				✓						✓						
S_3		✓				✓	✓				✓	✓												✓	
S_4	✓	✓	✓	✓	✓			✓	✓	✓	✓			✓		✓			✓		✓		✓	✓	
S_5					✓							✓						✓							
S_6	✓					✓		✓	✓	✓						✓	✓		✓	✓		✓			
S_7			✓	✓							✓			✓	✓			✓				✓			

by the obtained solutions. S_1 remains the best solution, connecting 17 of the 30 tested airports, and performing an index of 10 by 11, connecting 10 of the 11 best connected potentials corresponding to the new opened market.

S_1 solution provides 08 h 50 min maximum stopover time on the hub of Casablanca in outbound step, and 1 h 05 min (1 h 05 min) as minimal stopover time. Departure time is Thursday at 06:30 UTC from Casablanca, the return is the same day at 23:00, the inbound step is 55 min stopover time in casablanca and 10 h 40 min maximum before continuing correspondences back to connected airports. This study is applied to a real case of Moroccan national airline, and the results meet the expectations of policy makers. In the airline market between Africa and Europe, Royal Air Morocco has made a special place, far ahead of its North African counterparts and showing better prospects. Royal Air Morocco is distinguished from its competitors Maghreb in air links between Europe and Africa, and now allows you to tread on Air France's toes. The Moroccan company now ranks 2nd in the top 15 in terms of monthly passenger seats on the intercontinental segment, behind Air France. Obtained Departures from Casablanca fits perfectly the fleet programming strategy adopted by Royal Air Maroc company. Indeed, the company sends daily almost the two-thirds of its fleet on Africa within half an hour before midnight to return in the morning on the Casablanca hub before continuing to Europe, America and golf countries. And knowing that the best potentials of this example are African countries friends and allies of Morocco, the obtained arrival from the new opening around 23:00 (UTC) is perfectly convenient to the fleet departures to Africa starting around 23:30 (UTC). Similarly, a departure time toward the new opening at 06:30 (UTC) is very perfect knowing that the vast majority of those arrivals lands in Casablanca before 06:00(UTC).

21.4 Conclusions and Future Works

This paper provides a simple way to optimize the connectivity between several airports via a single hub airport for a given airline. Managers of airlines and policy-makers are likely interested in examining the results of this paper, that show how the planning of their fleets could be made without taking into account planned or unplanned connectivity. However, when receives a project to open a new destination, the reason may be political or purely commercial, we can thanks to this model, insert this new schedule without losing the planned connections and without causing important delays during correspondences, knowing that the delay is an annoying parameter and causes significant additional costs for the airline companies. Then this model will serve as a decision support tool for an air carrier for properly placing the new schedules in an already established airlines flight schedule. We have introduced new multi-objective optimization approach for the insertion of new routes in a given flight programme and this work is distinguished by the fact of being the first to introduce the problem of connectivity in the network of an airline whose main activity is based on hub structure. The pareto set is generated using SMP SO Algorithm and the numerical experiences are based on real instance of Royal Air Maroc schedule

on the hub of Casablanca. Finally, we are all convinced that airline industry is one of the most affected by operational disruptions, defined as deviations from originally planned operations. Due to airlines network configuration, delays are rapidly propagated to connecting flights, substantially increasing unexpected costs for the airlines [10, 11]. The goal in these situations is therefore to minimise the impact of the disruption, reducing delays and the number of affected flights, crews and passengers. As perspective of this chapter, we will include the disruption model in order to construct a robustness approach.

References

1. R. Abeyratne, Achieving competitive advantage through connectivity and innovation: an application in airline hubbing, in *Competition and Investment in Air Transport* (Springer International Publishing, Cham, 2016), pp. 131–143
2. G. Burghouwt, *Airline Network Development in Europe and Its Implications for Airport Planning* (Ashgate, Aldershot, 2007)
3. G. Burghouwt, R. Redondi, Connectivity in air transport networks: an assessment of models and applications. *J. Transp. Econ. Policy* **47**, 35–53 (2013)
4. K. Deb, D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms* (Wiley, New York, 2001)
5. N. Dennis, Competition between hub airports in Europe and a methodology for forecasting connecting traffic, in *World Transport Research: Selected Proceedings of the 8th World Conference on Transport Research*, vol. 1, 1999
6. C.V. Dyke, M. Meketon, B.W. Patty, in *Network Analysis and Simulation*, ed. by B.W. Patty. Handbook of Operations Research Applications at Railroads (Springer, New York, 2015), pp. 191–217
7. N. Ghaffari-Nasab, M. Ghazanfari, E. Teimoury, Robust optimization approach to the design of hub-and-spoke networks. *Int. J. Adv. Manuf. Technol.* **76**, 1091–1110 (2014)
8. Global Airport Connectivity Monitor, IATA Aviation Information and Research Department (2000)
9. D. Guimarans, P. Arias, M.M. Mota, in *Large Neighbourhood Search and Simulation for Disruption Management in the Airline Industry*, ed. by M.M. Mota, I.F.D.L. Mota, D.G. Serrano. Applied Simulation and Optimization (Springer International Publishing, Cham, 2015), pp. 169–201
10. R. Guimera, S. Mossa, A. Turtshi, L.A.N. Amaral, The worldwide air transportation network: anomalous centrality, community structure, and cities' global roles. *Proc. Natl. Acad. Sci.* **102**, 7794–7799 (2005)
11. D.-W.-I.L. Ionescu, D.C. Gwiggner, P.D.N. Kliewer, Data analysis of delays in airline networks. *Bus. Inf. Syst. Eng.* 1–15 (2015)
12. J. Kennedy, in *Particle Swarm Optimization*, ed. by C. Sammut, G.I. Webb. Encyclopedia of Machine Learning (Springer, New York, 2011), pp. 760–766
13. P. Malighetti, G. Martini, S. Paleari, R. Redondi, The Efficiency of European Airports: Do the Importance in the EU Network and the Intensity of Competition Matter? University of Bergamo, Bergamo, 2008
14. H. Rahil, B. Abou El Majd, A multi-objective optimization approach for hub connections. Application to the insertion of new flights in airline schedule, in *The 5th International Conference on Metaheuristics and Nature Inspired Computing, META'14*, Marrakech, 2014
15. H. Rahil, B. Abou El Majd, M. Bouchoum, Optimization of inserting a new flight in airline schedule using evolutionary algorithms, in *WISTL'14 Workshop Proceeding*, 2015

16. S.A. Taher, A. Karimian, M. Hasani, A new method for optimal location and sizing of capacitors in distorted distribution networks using PSO algorithm. *Simul. Model. Pract. Theory* **19**, 662–672 (2011)
17. J. Veldhuis, The competitive position of airline networks. *J. Air Transp. Manage.* **3**, 181–188 (1997)
18. M.D. Wittman, W.S. Swelbar, Modeling Changes in Connectivity at U.S. Airports: A Small Community Perspective. Report No. ICAT-2013-05 (2013)
19. C. Zhang, H. Shao, Y. Li, Particle swarm optimisation for evolving artificial neural network, in *2000 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4 (2000), pp. 2487–2490

Chapter 22

Solving the P/Prec, $p_j, C_{ij}/C_{max}$ Using an Evolutionary Algorithm

Dalila Tayachi

Abstract In this chapter, we tackle the problem of scheduling a set of related tasks on a set of identical processors taking into account the communication delays with the objective of minimizing the maximal completion time. This problem is well known as NP-Hard. As Particle swarm optimization PSO is a promising approach for solving NP-complete problems due to its simple implementation, fast convergence and its few parameters to adjust, the main contribution of this research is to use for the first time PSO to solve the multiprocessor scheduling problem with communication delays. The proposed approach HEA-LS is a hybrid algorithm involving particle swarm optimization PSO and local search algorithm LS. Experiments conducted on several benchmarks known in the literature prove the effectiveness of our approach and show that it compares very well to the state of the art methods.

Keywords Scheduling • Communication delays • Evolutionary algorithm • Hybridizing • Particle swarm • Local search

22.1 Introduction

The multiprocessor scheduling problem considered in this chapter deals with scheduling n tasks on m identical processors, in presence of communication delays. That means, in addition of classical precedence constraints between tasks, we must consider the communication delays when related tasks are processed on different processors. The objective of this scheduling problem is the minimization of the makespan. This problem is NP-Hard even in specific cases when the processing times and communication delays are unitary [12]. Only specific cases are solved in polynomial times when strong assumptions are made on the number of processors, the graph structure, or the communication times. For example, Chretienne [1] introduced an algorithm for the case of an unbounded number of processors, small communication delays and out-tree precedence constraints. Colin and Chretienne [2] proposed a polynomial algorithm for optimally scheduling in the case of an un-

D. Tayachi (✉)

Ecole Supérieure de Commerce de Tunis, Université de la Manouba, Manouba 2010, Tunisia
e-mail: dalilatayachi@gmail.com

bounded number of processors, small communication times and when duplication is allowed. Heuristics are proposed for solving general cases, like the algorithm of Hwang et al. [7], and the one of Wu and Gajski [15]. Metaheuristics are also provided for solving this problem such as the tabu search algorithm of Tayachi et al. [13], variable neighborhood search of Davidovic et al. [5], and the genetic algorithm of Omara and Arafa [11].

In this research, we propose an evolutionary algorithm based on particle swarm optimization to solve the multiprocessor scheduling problem with communication delays. The swarm is represented by permutations of n tasks, then ETF algorithm is applied to generate schedules.

The remainder of this chapter is structured as follows: first we give the notation of the problem, then we provide related works, next we present a brief description of PSO algorithm followed by our approach. Finally, we give results of experimental tests carried on different benchmark problems.

22.2 Problem Formalization

Given a set of n non preemptive tasks $I = \{1, \dots, n\}$. Each task is characterized by its processing time p_i , a set of successors $S(i)$ and a set of predecessor $P(i)$. These tasks constitute an acyclic directed and valuated graph $G = (I, U, P, C)$ in which a node correspond to a task and an arc (i, j) represent the precedence constraints. $\eta(i, j)$ is a positive integer which represents the number of messages sent from i to j where i is an immediate predecessor of j . A set of m identical processors $\{\Pi_1, \dots, \Pi_m\}$ is available. A parameter $d(\Pi_i, \Pi_j)$ is introduced to represent the transfer of a unit time from processor Π_i to processor Π_j , that is if an arc (i, j) in U , then the time taken between i and j is $C_{ij} = \eta(i, j) * d(\Pi_i, \Pi_j)$. If i and j are performed on the same processor then $C_{ij} = 0$.

A schedule S of the problem consists of assigning to each task i a starting time t_i and a processor Π_i in such a way that:

- For any arc (i, j) of G , $t_j \geq t_i + p_i$ if $\Pi_i = \Pi_j$ and $t_j \geq t_i + p_i + c_{ij}$ if $\Pi_i \neq \Pi_j$
- At any time, each processor executes at most one task; and m processors are busy.

The objective is to determine a schedule whose makespan C_{max} is minimum. Following the notation of Veltman [14] this problem is denoted by P/Prec, $p_j, C_{jk}/C_{max}$.

Figures 22.1 and 22.2 show respectively a precedence graph with communications delays and the corresponding schedule produced on two processors taking into account precedence constraints and communication delays. For example, task 2 precedes task 3 and the value of the precedence arc between them is three units. As task 3 and task 2 are assigned to different processors (Fig. 22.2) task 3 cannot start before a communication delay of three units passed after the end of task 2.

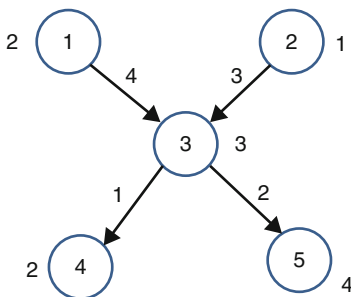


Fig. 22.1 Precedence graph with communication delays

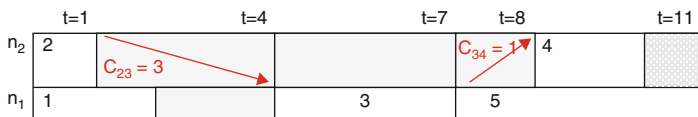


Fig. 22.2 A schedule of the previous precedence graph on two identical processors taking into account the communication delays, $C_{max} = 11$

22.3 Related Work

The main exact methods devoted to solve the general case of multiprocessor scheduling problem with communication delays are the branch and bound method of Daddi-Moussa [3] and another branch and bound algorithm proposed by Kwok and Ahmad [9] in the case of fully connected multiprocessor system. However, due to the computational complexity of this scheduling problem these approaches solve only instances of small size. The best known heuristics for solving the general case of this problem are list algorithms. These algorithms allocate tasks into the available processors without back tracking. We emphasize ELS (*Extended list Schedule*) of Hwang et al. [7]. This heuristic adopts a two phase strategy: in the first communication delays are ignored and a solution is determined by a classical list algorithm and in the second step, communication delay is added to the schedule obtained in the first step. The same author provided a second algorithm ETF (*Earliest Task First*) with better performance. It's a generalization of the list algorithm of Smith (1987) in the case of unitary processing times and unitary communication delays. This list algorithm assigns first the earliest ready task. It offers the possibility to postpone a scheduling decision to the next decision time if a task completion occurring between two successive decision times makes a more urgent task schedule. Wu and Gajski [15] computes, in the modified critical path MCP algorithm, for each task a list priority. This priority is the longest path from the task and an end node including communication delays. Then the task with the highest priority is assigned to a processor that allows the earliest start time.

Metaheuristics have been also used in solving the multiprocessor scheduling problem with communication delays $P/Prec, p_j, C_{jk}/C_{max}$ but the number of papers found in the literature remain little. The most significant and known are a parallel genetic algorithm proposed by Kwok and Ahmad [10]. Authors evaluated their algorithm through a comparison with two heuristics using random task graph for which optimal solutions are known. Tayachi et al. [13] proposed a tabu search algorithm operating in two phases based on modeling the problem by a disjunctive graph. They studied the cases of small and large communication delays. Experiments carried on arbitrary graphs and comparison with many heuristics show the efficiency of the method. Davidovic et al. [5] provided a variable neighborhood search and illustrate the performance of their algorithm by testing it on randomly generated task graphs. Notice that in most of algorithms developed, experimental results are reported by applying them on new set instances only, making difficult a fair comparison between them. Recently, Davidovic and Crainic [4] proposed new sets of benchmark instances. They developed several algorithms such as variable neighborhood method, a genetic algorithm and a local search and they applied them on these benchmarks in order to investigate the characteristics of the proposed test-problem instances.

Thus, the first objective of this paper is to provide for the first time a comparison between different metaheuristics for the multiprocessor scheduling problem with communication delays by carrying experiments on these benchmarks. More recently, Omara and Arafa [11] proposed a genetic algorithm based on critical path heuristic. The evaluation process was only provided using standard task graphs with random communication delays.

The second objective of this work is to provide a new evolutionary algorithm based on particle swarm optimization to solve the multiprocessor scheduling problem with communication delays. To the best of our knowledge this is the first application of the PSO algorithm for this problem. The main elements of this algorithm will be described in the next section.

22.4 A New Hybrid Evolutionary Algorithm Based on Particle Swarm Optimization HEA-LS

22.4.1 Particle Swarm Optimization

Particle swarm optimization is a metaheuristic introduced by Kennedy and Eberhart [8] simulating swarming habits of animals like birds and fish in the nature. It is based on a population of particles moving in the search space and each one is a potential solution. Each particle memorizes information about its best solution visited and the best solution known in its neighborhood. A basic iteration in this algorithm consists

of updating the position of each particle in the swarm in order to move to a new position according to these equations:

$$V_i^{t+1} = w.V_i^t + r_1.c_1(lbest_i - x_i^t) + r_2.c_2(Gbest_i - x_i^t) \quad (22.1)$$

$$x_i^{t+1} = x_i^t + V_i^{t+1} \quad (22.2)$$

In these equations, x_i^t indicates the position of a particle at time t in the search space; V_i^t its velocity; $lbest_i$ the best solution visited by the particle and $Gbest_i$ is the best solution in the whole swarm. Three parameters c_1 , c_2 and w are respectively cognitive factor, social factor and the inertia. The scheme of this algorithm is summarized Fig. 22.3.

Basic PSO algorithm

```

Generate the initial swarm ;
Find the Gbest ;
Nbr_Gen ← 1 ;
While Nbr_Gen < Max_Gen do
  For each particle P of the Swarm do
    Update the position;
    Evaluate the position
    Update the Lbest ;
  End for
  Update Gbest ;
  Nbr_Gen ← Nbr_Gen+1
End while

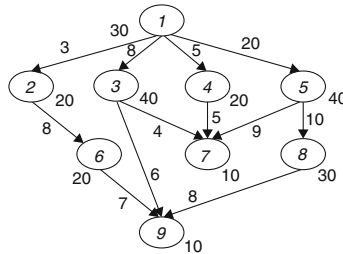
```

Fig. 22.3 Basic PSO algorithm

22.4.2 Position Representation and Fitness Evaluation

In our Approach, a position is a permutation of n tasks. The initial population is generated randomly. For each particle is associated an $lbest_i$ which is also a randomly generated permutation. For each permutation is applied the ETF algorithm to produce the schedule. The fitness value of each particle is then the length of the schedule C_{max} obtained by ETF.

Figure 22.4 shows an illustrative example of multiprocessor scheduling with a number of processors $m = 2$, a precedence graph of nine tasks and a swarm of four particles. C_{max_p} is the length of the schedule obtained by ETF (Fig. 22.4).



(a) Precedence Graph

Population		
Particle	$Cmax_p$	current position (pos)
1	135	6 8 9 1 2 4 3 5 7
2	149	2 7 1 4 8 9 3 6 5
3	145	2 6 1 4 3 9 7 8 5
4	150	8 9 7 3 6 2 4 5 1

$Cmax_p$ = $Cmax$ obtained by applying ETF Algorithm on the current position (pos)

Fig. 22.4 Illustration of particle position representation for (a) the precedence graph. (b) $Cmax_p$ = $Cmax$ obtained by applying ETF Algorithm on the current position (pos)

22.4.3 Position Update

At each iteration, the position of each particle is updated by a recombination of the current position, the best position of the particle and the global best of the whole swarm. This recombination is based on the PSO proposed by Dang et al. [6] for the Team Orienteering Problem TOP, and is interpreted as an extraction of a number of tasks from each position. The number of tasks extracted from each position are $w \cdot n$ from the current position, $(1-w) \cdot n \cdot c_1 \cdot r_1 / c_1 \cdot r_1 + c_2 \cdot r_2$ from local best position and $(1-w) \cdot n \cdot c_2 \cdot r_2 / c_1 \cdot r_1 + c_2 \cdot r_2$ from the best neighbor in the swarm. This extraction is illustrated in Fig. 22.5.

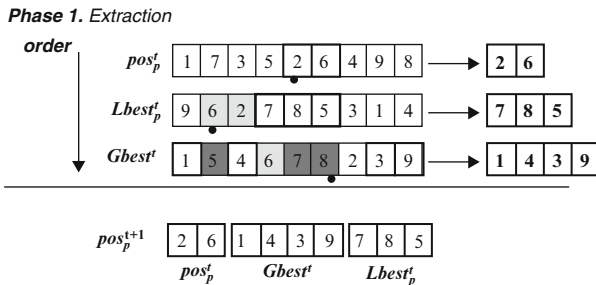


Fig. 22.5 Example of position update

In this example, the PSO parameters are set as follows: $w=0.25$, $c_1=0.5$ and $c_2 = 0.4$. r_1 and r_2 are equal to 0.5. Thus the number of tasks extracted from each position are respectively 2 ($= 0.25 * 9$), 3 ($= (1 - 0.25) * 9 * 0.5 * 0.5 / (0.5 * 0.5 + 0.4 * 0.5)$), 2 ($= \lfloor 0.25 * 9 \rfloor$), 3 ($= \lfloor (1 - 0.25) * 9 * 0.5 * 0.5 / (0.5 * 0.5 + 0.4 * 0.5) \rfloor$), 4 ($= 9 - 2 - 3$). Let note that our PSO algorithm is a discrete variant of PSO and classified as a PSO with position that no velocity is used.

22.4.4 The Hybrid HEA-LS Algorithm

In order to enforce the effectiveness of our discrete evolutionary algorithm we opt to integrate two local search procedures: a local search on the positions and a local search on the schedules.

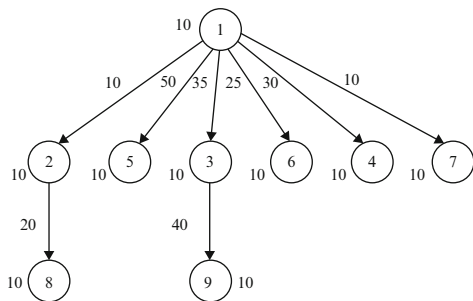
22.4.4.1 Local Search on Positions

In this procedure each new position has a probability to be improved by one of shift or swap operator. The shift operator consists at moving each task from its position to any other position. In the swap operator, every two successive tasks are exchanged in the permutation. Local search on positions allows us to explore more permutations in the search space in order to obtain new permutation giving better C_{max} . Given that the evaluation of each permutation is done by the list algorithm ETF which didn't ensure optimality, we decide to apply local search on schedule (solution).

22.4.4.2 Local Search on Schedules

When a new local best is found we apply local search on the schedule obtained. This local search is based on the graph decision model [10]. In this model, in addition to precedence constraints, we take into consideration additional constraints expressing relative order of independent tasks on the same processor. Thus, an instance of the problem P/Prec, $p_j, C_{ij}/C_{max}$ is modeled by a decision graph $G_d = (I, U \cup A)$ where U is the set of precedence constraints and A the set of decision arcs. A decision arc (i, j) belongs to A if tasks i and j are independent tasks and sequenced successively on the same processor. This decision graph is acyclic and the length of the schedule is the length of the critical path in G_d . Thus, to determine the optimal schedule we should determine the decision arcs set such that the critical path of G_d (A) is minimum.

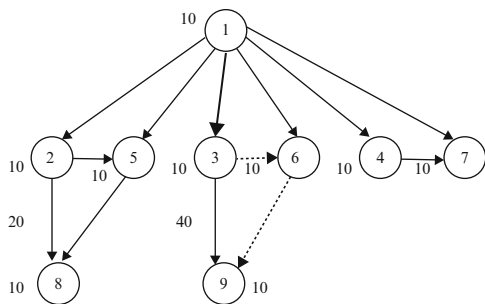
In Fig. 22.6, we illustrate the graph decision model of a multiprocessor scheduling problem with three processors and nine related tasks. (Dotted arcs are the decision arcs.)



(a) Precedence graph

	10	20	30	40	50	60	70	75
π_1	1	2	5	8				
π_2					3	6	9	
π_3				4	7			

(b) Schedule of the precedence graph (a) with $m=2, C_{max}=75$



(c) A decision graph of a P3/Prec, $P_j, C_{ij}/C_{max}$

Fig. 22.6 Decision graph of a multiprocessor scheduling problem with nine tasks and two homogenous processors [10]. (a) Precedence graph. (b) Schedule of the precedence graph (a) with $m = 2, C_{max} = 75$. (c) A decision graph of a P3/Prec, $p_j, C_{ij}/C_{max}$

Based on this model, the candidate tasks to move are the critical ones. Our local search contains three neighborhoods proved to be efficient in our preceding work [10]:

- In the first neighborhood, each critical task is moved from its processor to the minimal load processor in order to achieve a load balance between processors.
- In the second, we permute every two critical tasks that are independent and sequenced successively on the same processor.
- Finally, the third neighborhood is built by assembling each pair of critical tasks on the same processor.

Our local search is a descent method that is at each iteration, we restart with the best neighbor.

22.5 Performance Comparison and Results

We evaluate the performance of our HEA-LS algorithm by carrying several experimental tests on different benchmark problems. The first set benchmark instances are from Kwok and Ahmad [9] in which the processors are assumed to be fully connected. These problems are divided into three classes. The first one denoted RGBOS contains 36 random graphs with Branch and bound obtained optimal solutions. The number of tasks n ranges between 10 and 32 and $m = 2$. The second class called RGPOS is composed of 30 instances with known optimal solutions in which the number of tasks n ranges from 50 to 500 and m ranges between 3 and 11 processors. The third class denoted RGNOS contains 250 random task graphs, with n ranging between 50 and 500, and the optimal solutions are unknown. The second set of benchmark instances is from Davidovic and Crainic [4], in which fully connected and hypercube multiprocessor are considered.

HEA-LS is coded in C++ using Standard Template Library STL for data structure the program is compiled with GNU GCC in Microsoft Windows environment and the tests were conducted *Intel Core 2 Duo (2.2 GHz)* processor.

The parameters set for the proposed algorithms are $c1 = c2 = 0.5$, $r1$ and $r2$ are randomly generated in $[0, 1]$, the size of the swarm is proportional to the number of tasks n and termination criterion used is the total number of iterations which ranges from 500 to 5000.

By applying HEA-LS algorithm on all the instances of RGBOS and RGPOS, we have found optimal solution. For the RGNOS class, the optimal solutions are unknown. In [3], the authors give solutions by applying many heuristic methods on 50 graphs of this class in which the number of tasks ranges from 50 to 500, and a number of processors is equal to 2. Our algorithm, HEA-LS gives in almost 80% of cases better results than all the heuristics cited in [3]. Results are reported in the tables below.

In Tables 22.1, 22.2, 22.3, and 22.4, the first column contains the names of the instances, the second column gives the number of the processors and the column 3 reports the results obtained by our algorithm HEA-LS. The remainder columns contain the results obtained by many heuristics proposed in [3].

Table 22.5 displays comparison results of HEA-LS and the other methods of Davidovic and Crainic [4] over 50 instances (five graphs for each n). $Av.C_{max}$ denotes the average of C_{max} . From this table, we observe that HEA-LS outperforms the other methods.

We have also applied our algorithm on 50 instances of the benchmarks proposed in [3]. These instances are randomly generated test instances with known optimal solutions. The number of tasks n ranges between 50 and 500. The authors give results of different heuristics and metaheuristics such as Variable neighborhood VNS, two genetic algorithms PSGA, GA, Local search LS and Critical Path CP.

Table 22.6 compares the results obtained by the hybrid proposed algorithm HEA-LS with the results given in [3] in the case of 4 processors. Column 2 contains the optimal length of the schedule SL_{opt} , Other columns give the average percentage deviations from the optimal solutions. It is clear that the HEA-LS performs better than PSGA, LS and CP, and is closer than GA but VNS gives the best results.

Table 22.1 Scheduling results for RGBOS instances

Graph	p	$Opt.C_{max}$				C_{max}	
		<i>HEA-LS</i>	<i>CPES LPTES</i>	<i>LBMC</i>	<i>DC</i>	<i>LPTDC</i>	<i>PPS</i>
r10_0	2	271	271	488	352	275	284
r10_1	2	207	207	281	279	283	289
r10_10	2	266	266	1342	902	902	1163
r12_0	2	307	307	522	395	339	414
r12_1	2	246	246	366	320	356	352
r12_10	2	232	232	916	780	701	970
r14_0	2	281	281	428	339	326	399
r14_1	2	271	271	629	370	368	373
r14_10	2	267	267	1212	843	882	678
r16_0	2	332	332	441	485	373	428
r16_1	2	326	326	620	358	428	489
r16_10	2	416	416	2175	1040	1109	2517
r18_0	2	420	420	815	512	481	569
r18_1	1	428	428	892	476	732	526
r18_10	2	390	390	1860	1201	1255	2147
r20_0	2	477	477	962	566	547	579

Table 22.2 Scheduling results for RGBOS instances

Graph	p	$Opt.C_{max}$				C_{max}	
		<i>HEA-LS</i>	<i>CPES LPTES</i>	<i>LBMC</i>	<i>DC</i>	<i>LPTDC</i>	<i>PPS</i>
r20_1	2	378	378	783	469	521	596
r20_10	2	457	457	3361	1249	1249	2989
r22_0	2	585	585	1023	686	708	784
r22_1	2	488	488	905	488	605	996
r22_10	2	575	575	2567	1292	1356	3370
r24_0	2	480	480	882	634	557	625
r24_1	2	618	618	1276	800	817	961
r24_10	2	594	594	3843	1349	1372	2691
r26_0	2	737	737	1319	893	781	934
r26_1	2	614	614	1097	687	687	1150
r26_10	2	529	529	909	1128	1196	4287
r28_0	2	680	680	1269	750	731	958
r28_1	2	671	671	1194	875	312	1279
r28_10	2	623	623	4507	1467	1467	3189
r30_0	0	750	750	1435	849	790	1068
r30_1	2	811	811	1842	859	1144	1530
r30_10	2	653	653	3519	1388	1333	4692
r32_0	2	749	749	1337	894	821	1186
r32_1	2	886	886	1806	886	1353	1698
r32_10	2	941	941	5487	1675	1713	6453

Table 22.3 Scheduling results for RGPOS instances

Graph	p	$Opt.C_{max}$			C_{max}		
		<i>HEA-LS</i>	<i>CPES LPTES</i>	<i>LBMC</i>	<i>DC</i>	<i>LPTDC</i>	<i>PPS</i>
o50_0	3	933	933	2085	1690	1473	1864
o50_1	3	956	956	2403	1807	1790	2176
o50_10	3	811	811	6166	2281	2360	7255
o100_0	5	898	898	2601	2435	1939	1501
o100_1	5	831	831	2583	2353	2032	1704
o100_10	5	929	929	7138	4602	4650	7349
o150_0	6	1215	1215	3442	3973	3166	2253
o150_1	6	1104	1104	3490	3464	3342	2539
o150_10	6	1186	1186	11,269	6583	6319	11,879
O200_0	7	1351	1351	4807	4895	4091	3175
O200_1	7	1345	1345	5343	4663	4450	3336
O200_10	7	1446	1446	18,640	8824	8473	17,132

Table 22.4 Scheduling results for RGPOS instances

Graph	p	$Opt.C_{max}$			C_{max}		
		<i>HEA-LS</i>	<i>CPES LPTES</i>	<i>LBMC</i>	<i>DC</i>	<i>LPTDC</i>	<i>PPS</i>
O250_0	7	2553	2553	9505	9696	9855	8532
O250_1	7	2377	2377	9399	8750	8448	7127
O250_10	7	2357	2357	27,198	12,238	12,835	26,373
O300_0	8	2464	2464	8798	10,316	9601	5850
O300_1	8	2250	2250	10,388	9831	3098	6520
O300_10	8	2397	2397	28,860	14,076	14,767	23,844
O350_0	9	2342	2342	8986	11,177	9880	5299
O350_1	9	2371	2371	9582	11,012	10,590	7879
O350_10	9	2409	2409	28,824	16,906	16,855	24,231
O400_0	10	1796	1796	6218	9258	7387	3809
O400_1	10	1798	1798	7290	8854	7504	4554
O400_10	10	1781	1781	24,140	13,889	14,626	20,964
O450_0	10	2763	2763	10,363	15,207	12,273	6597
O450_1	10	2872	2872	12,478	14,645	14,470	9924
O450_10	10	3168	3168	42,928	22,035	23,214	33,480
O500_0	11	2085	2085	7356	11,864	9744	4663
O500_1	11	2050	2050	9288	11,041	9789	5488
O500_10	11	2054	2054	28,111	16,049	17,545	24,285

Table 22.5 Scheduling results for RGNOS instances

<i>n</i>	Av. C_{max}						
	<i>HEA-LS</i>	<i>CPES</i>	<i>LPTES</i>	<i>LBMC</i>	<i>DC</i>	<i>LPTDC</i>	<i>PPS</i>
50	1214	1232.6	1328	1838	1398	1379.4	1573.6
100	2478.6	2530.4	2572.4	3907	2699.6	2936.4	3106.2
150	3782.8	3861.6	3962.2	5749	4164.2	4770.4	4875
200	5013	5105.6	5283	7339.2	5497.4	6373	6014.6
250	6433.4	6602.4	6750	10,134.4	6863.6	8333	7553.4
300	7626.2	7675.6	7882	11,446.2	8055.2	9462.8	8887.2
350	8753.6	8819.4	8989.2	13,164.2	9215	10,403	10,441
400	10,123.2	10,301.2	10,585.2	14,975.6	10,645.6	12,446.2	12,088.4
450	11,309.8	11,523.8	11,767.8	16,406.2	11,936	13,353.6	13,226.2
500	12,476.6	12,562.8	12,811.4	17,928.6	13,249	15,618	14,531.4

Table 22.6 Scheduling results for benchmarks of Davidovic and Crainic [4] with optimal known solutions and $m = 4$

<i>n</i>	SL_{opt}	Av. percentage deviations					
		<i>HEA-LS</i>	<i>CP</i>	<i>LS</i>	<i>VNS</i>	<i>GA</i>	<i>PSGA</i>
500	600	16.70	48.2	22.3	2.42	15.7	12.43
100	800	24.29	57.54	36.22	10.14	25.46	35.37
150	1000	59.65	74.7	38.96	11.47	44.33	42.5
200	122	58.68	86.97	59.78	24.5	49.47	35.95
250	1400	59.23	78.48	61.93	31.05	65.69	46.53
300	1600	64.74	82.82	66.17	53.43	72.92	78.99
350	1800	40.33	82.24	61.15	35.51	63.52	72.88
400	200	68.37	83.66	80.96	47.21	26.37	77.99
450	2200	65.30	85.36	58.85	31.76	30.14	56.69
500	2400	46.46	84.87	38.59	37.45	76.96	69.76
Average		50.37	76.49	52.49	28.5	47.06	52.91

To improve the performance, further experiments should be carried to adjust the various parameters of the HEA-LS, and an investigation of new neighborhood should be done especially for the hypercube multiprocessor architecture.

22.6 Conclusion and Perspectives

In this chapter, we proposed a hybrid evolutionary algorithm for the multiprocessor scheduling problem with communication delays $P/Prec, p_j, C_{jk}/C_{max}$. This problem is a difficult and important scheduling problem due to the assumption that communication delays between tasks are not negligible. Our approach is based on particle swarm optimization and the modeling of the problem by a decision graph. In numerical results, we used benchmarks of Davidovic and Crainic [4] and we compared our algorithm with various heuristics and metaheuristics. Experiments show the good

performance of the addressed algorithm HEA-LS. To the best of our knowledge, the PSO is applied to this scheduling problem for the first time in this research.

For further work, some perspectives could be considered. Firstly, in addition to experiments conducted on complete interconnection network it would be interesting to evaluate performance of the proposed algorithm on other multiprocessor architecture systems like the hypercube. Secondly, an interesting perspective could be the modification of our algorithm to solve the case of the heterogeneous multiprocessor. Lastly, we will intend to consider other objective criteria in this multiprocessor scheduling problem.

Acknowledgements I would like to thank Mr. Aziz Moukrim professor at Université de Technologie de Compiègne UTC in France for his helpful remarks and suggestions to improve this work.

References

1. P. Chretienne, A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints. *Eur. J. Oper.* **43**, 225–230 (1989)
2. J.-Y. Colin, P. Chretienne, CPM scheduling with small communication delays. *Oper. Res.* **39**, 680–684 (1995)
3. A. Daddi-Moussa, Méthode exacte pour les problèmes avec délais de communication. Thèse de Doctorat de l'Université de Paris VI (1997)
4. T. Davidovic, T. Crainic, Benchmark-problem instances for static scheduling of task graphs with communication delays on homogeneous multiprocessor systems. *Comput. Oper. Res.* **33**, 2155–2177 (2006)
5. T. Davidovic, P. Hansen, N. Mladenovic, Variable neighborhood search for the multiprocessor scheduling problem with communication delays, in *MIC 2001*, Porto (2001)
6. D.-C. Dang, R.N. Guibadj, A. Moukrim, An effective PSO inspired algorithm for the team orienteering problem. *Eur. J. Oper. Res.* **229**, 332–344 (2013)
7. J.J. Hwang, Y.C. Chow, F.D. Angers, C.Y. Lee, Scheduling graphs in systems with interprocessor communication times. *J. Comput.* **18**, 244–257 (1989)
8. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceeding of IEEE International Conference on Neural Networks* (1995), pp. 1942–1948
9. U. Kwok, I. Ahmad, Efficient scheduling of arbitrary task graphs to multiprocessors using a parallel genetic algorithm. *J. Parallel Distrib. Comput.* **47**, 58–77 (1997)
10. U. Kwok, I. Ahmad, Benchmarking and comparison of the task graph scheduling algorithms. *J. Parallel Distrib. Comput.* **59**, 381–422 (1999)
11. F. Omara, M. Arafa, Genetic algorithms for task scheduling problem. *J. Parallel Distrib. Comput.* **70**, 13–22 (2010)
12. V.J.R. Smith, UET scheduling with Unit interprocessor communication delays. *Discret. Appl. Math.* **18**, 55–71 (1987)
13. D. Tayachi, P. Chretienne, K. Mellouli, Une méthode tabou pour l'ordonnancement multiprocesseur avec délais de communication. *RAIRO Oper. Res.* **34**, 467–485 (2000)
14. B. Veltman, B.J. Lageweg, J.K. Lenstra, Multiprocessor scheduling with communication delays. *Parallel Comput.* **16**, 173–182 (1990)
15. M. Wu, D. Gajski, Hypertool: a programming aid for message passing systems. *IEEE Trans. Parallel Distrib. Syst.* **1**(3), 330–343 (1990)

Chapter 23

A User Experiment on Interactive Reoptimization Using Iterated Local Search

David Meignan

Abstract This article presents an experimental study conducted with subjects on an interactive reoptimization method applied to a shift scheduling problem. The studied task is the adjustment, by a user, of candidate solutions provided by an optimization system in order to introduce a missing constraint. Two procedures are compared on this task. The first one is a manual adjustment of solutions assisted by a software that dynamically computes the cost of the current solution. The second procedure is based on reoptimization. For this procedure, the user defines some desired changes on a solution, and then a reoptimization method is applied to integrate the changes and reoptimize the rest of the solution. This process is iterated with additional desired changes until a satisfactory solution is obtained. For this interactive approach, the proposed reoptimization procedure is an iterated local search metaheuristic. The experiment, conducted with 16 subjects, provides a quantitative evaluation of the manual and reoptimization approaches. The results show that, even for small local adjustments, the manual modification of a solution has an important impact on the quality of the solution. In addition, the experiment demonstrates the efficiency of the interactive reoptimization approach and the adequacy of the iterated local search method for reoptimizing solutions. Finally, the experiment revealed some limitations of interactive reoptimization that are discussed in this article.

Keywords Interactive optimization • Shift scheduling • Heuristic • Reoptimization

23.1 Introduction

Optimization-based decision support systems are tools that can support a decision maker to solve a complex optimization problem [15]. Such a system provides the decision maker with at least a model of the optimization problem, an optimization procedure for solving it, and means of instantiating the optimization model as well as analyzing solutions [4]. However, in many cases, the optimization model does not

D. Meignan (✉)

Universität Osnabrück, Institut für Informatik, Albrechtstraße 28, 49069 Osnabrück, Germany
e-mail: dmeignan@uni-osnabrueck.de

capture all aspects of the real problem. For instance, some aspects of the problem may be too difficult to express in mathematical terms (e.g. a robustness criterion), or the optimization model may have been simplified for being tractable. This gap between the real problem and the optimization model can result in the computation of inadequate or unrealistic solutions. In such cases, the decision maker can adjust the solutions to introduce the missing aspects. The adjustment can be performed manually which can be complex and potentially impair the quality of the solutions. Alternatively, the adjustment process can be done interactively using a reoptimization procedure that introduces local modifications in a solution while maintaining the quality of the entire solution [10].

Reoptimization procedures are used in two different contexts, namely dynamic optimization and interactive optimization. In dynamic optimization, the problem data change over time and a reoptimization procedure can be used to adapt a solution according to the perturbations (see for instance [14, 2, 16]). In interactive optimization, the perturbations do not correspond to modifications of the real problem, but are adjustments of a solution requested by a user of the optimization system. Our study investigates the latter context, where a reoptimization procedure is used in an interactive process for adjusting a solution. We use the term *interactive reoptimization* for this process [12].

In the research literature there are relatively few studies on interactive reoptimization. In [13] the author reviews different optimization methods for scheduling problems and suggests the use of a reoptimization procedure when a user modifies a solution. In the interactive reoptimization method proposed in [13], the user can edit manually a solution and then reoptimize the solution while keeping the manually modified parts *frozen*. Another interactive reoptimization approach is proposed in [5] for solving linear optimization problems. The authors study in particular the *stability* and *responsiveness* of different reoptimization algorithms. The stability is the ability of a reoptimization procedure to minimize the changes it induced on the initial solution. The responsiveness is related to the computation time required for reoptimizing a solution. Finally, in [10] an interactive reoptimization method is proposed for a shift scheduling problem. The work presented in the current paper is based on this latter study.

To the best of our knowledge, no experimental study with real users on interactive reoptimization is reported in the research literature. Results presented in previous works, such as in [5] and [10], are computational evaluations of reoptimization procedures. It should be noted that for other interactive optimization approaches, such as human-guided search [8], experimental studies with test subjects have been performed (see for instance [1, 8]) but the purpose of the interaction is different from that of interactive reoptimization. Looking at the interactive aspect of reoptimization, it appears important to validate such an approach through an experimental study with real users, which is the principal objective of the work presented in this article. The proposed experiment quantitatively evaluates an interactive reoptimization approach with real interactions and realistic datasets. In addition, we compare the reoptimization approach with manual edition in order to determine the gain of interactive reoptimization for adjusting solutions. Finally, we identify the

limitations of the proposed interactive reoptimization approach by examining the gap between reoptimized and ideal solutions. In summary, the presented experiment aims at providing experimental evidences for promoting the use of interactive reoptimization and also aims at encouraging further experimental investigations of interactive optimization methods.

This paper is a revised version of [11] presented at the Metaheuristics International Conference in 2015.

In the next section, the interactive reoptimization process is detailed and the optimization problem on which it is applied is presented. In Sect. 23.3 the goals and the method of the experimental evaluation are explained. The results of the experiment are presented and discussed in Sect. 23.4. Finally, concluding remarks are given in Sect. 23.5.

23.2 Interactive Reoptimization for Shift Scheduling

Interactive reoptimization aims at adjusting solutions when inaccuracies in an optimization model result in inadequate computed solutions. Although the enrichment of an optimization model is preferable to the adjustment of solutions, in many cases it is not possible to integrate all aspects of the real problem in an optimization model. In this context, it is necessary to rely on the user to adjust solutions according to the real problem. The interactive reoptimization approach therefore assumes that the user has an expertise in the application domain, but in turn, it does not require knowledge in modelling or optimization. In this section we detail the interactive reoptimization approach that has been proposed for a shift scheduling problem.

23.2.1 *Interactive Process*

The proposed interactive reoptimization process starts with an initial solution computed by an optimization procedure. We consider the case where this initial solution has been computed with an incomplete or inaccurate optimization model and consequently may require some modifications. The solution is presented to the user who can check whether the solution is valid or needs to be adjusted. If some adjustments have to be made on the solution, the user specifies the changes that are required and runs a reoptimization procedure for integrating them. In the proposed reoptimization approach, the changes requested by the user are specified in terms of preferred values on decision variables (i.e. a set of preferred or inadequate values can be assigned to each decision variable). The reoptimization procedure aims at integrating the requested changes and reoptimizing globally the solution in order to maintain the quality of the solution. Since the reoptimization can modify some parts of the solution where no changes have been requested, it may be necessary to per-

form several iterations of reoptimization with additional changes' requests before obtaining a satisfactory solution. For these subsequent iterations, the last reoptimized solution is used as the starting solution and additional requested changes are combined with previous ones to avoid recurrence of inadequate components in the solution.

The main computational component of this interactive process is the reoptimization procedure. In comparison to the optimization procedure used for generating the initial solution, the reoptimization procedure has particular features. First, in addition to the initial constraints and objectives, the optimization model used for the reoptimization contains two supplementary objectives. The first one aims at integrating the changes requested by the user. In the proposed reoptimization model, this first additional objective ensures that decision variables are set with preferred values (when such a preference exists). The second additional objective minimizes the distance between the initial solution and the solution reoptimized. This objective ensures the stability of the reoptimization which is essential for the convergence of the interactive process toward a satisfactory solution. Besides these differences between the optimization model and the reoptimization model, the requirements in terms of computation time are more important for reoptimization than for the initial optimization process. Due to the interaction, the reoptimization should only take a few seconds to keep the user focused. To meet this requirement, the initial solution can be used as a starting point for the reoptimization. However, it should be noted that even with a good initial solution the complexity of a reoptimization problem generally remains the same as the initial optimization problem [2].

23.2.2 Shift Scheduling

The proposed interactive reoptimization approach has been evaluated on a staff scheduling problem. This application domain is promising for applying interactive reoptimization and more generally it is an interesting domain for interactive optimization approaches due to the difficulty in modeling and solving the related optimization problems [3]. The fact that the solutions to staff scheduling problems directly impact employees' activities reinforces the need for interactions between the decision maker and the optimization system in order to adjust and validate solutions. In a real context, an optimization model for staff scheduling is likely to present some simplifications and omissions which could be solved using an interactive reoptimization approach.

The shift scheduling problem consists in assigning shifts (e.g. Early shift 7:00–15:00, Late shift 15:00–22:00) to employees for a given planning period. The result of the optimization is a roster defining the schedule of each employee. The constraints retained for the problem model are drawn from a problem proposed for the International Nurse Rostering Competition held in 2010 (INRC2010) [7]. The initial INRC2010 model contains about 20 different constraints but it has been simplified for the purpose of the experiment.

The problem model used for the generation of initial solutions contains three types of constraints, namely a hard constraint (H), work regulations (R_1 – R_5), and soft constraints (S_1 – S_5). The hard constraint ensures that the roster is complete:

H (*Complete roster*) For each day in the planning horizon, the number of employees assigned to a shift must be equal to the demand. An employee can only have one shift assigned per day.

Work regulations are requirements specified by the contracts of the employees and that must be satisfied in a roster:

R_1 (*Maximum number of assignments*) The total number of assignments of an employee within the whole planning period must not exceed a given maximum value.

R_2 (*Minimum number of assignments*) The total number of assignments of an employee within the whole planning period must not be less than a given minimum value.

R_3 (*Maximum number of consecutive working days*) The number of consecutive working days of an employee must not exceed a given maximum value.

R_4 (*Incompatible shift sequences*) Some pairs of shifts cannot be worked on two consecutive days. For instance, an *Early* shift cannot be worked the day after an *Night* shift.

R_5 (*Days-off*) No shift must be assigned to an employee for the days he has requested days-off.

Soft constraints are rules for improving the quality of employees' schedule and are not mandatory:

S_1 (*Minimum number of consecutive working days*) The number of consecutive working days of an employee should not be less than a given minimum value.

S_2 (*Maximum number of consecutive days-off*) The number of consecutive days-off of an employee should not exceed a given maximum value.

S_3 (*Minimum number of consecutive days-off*) The number of consecutive days-off of an employee should not be less than a given minimum value.

S_4 (*Complete weekends*) Weekends should be either entirely worked or completely free.

S_5 (*Identical shifts during weekend*) During weekends completely worked, an employee should have the same shift assigned.

As we mentioned in the previous section, two constraints are added to the problem model for the reoptimization procedure. The first constraint (P) aims at integrating the changes requested by the user, the second constraint (D) minimizes the distance between the initial solution and the reoptimized solution.

P (*Preferences on assignments*) For adjusting a solution the user can specify preferred assignments (e.g. the user can indicate that *Early shift* and *Day-off* are preferred assignments for a given day and employee). A preference is satisfied when the assignment corresponds to one of the preferred assignments.

D (Distance to initial solution) The distance constraint penalizes any deviation from the initial solution that is reoptimized. An assignment in the solution that is different to the initial assignment corresponds to a penalty of one.

For the optimization and the reoptimization procedures, an order between these different sets of constraints is defined. The global objective, reported in Eq. (23.1), is the minimization of the number of unsatisfied constraints using a lexicographic order. This means that the satisfaction of the hard constraint (H) takes priority over all other constraints. Then, the satisfaction of work regulations (R_1 – R_5) comes in second rank. If some preferences on assignments (P) have been defined by the user, they are less important than work regulations but have priority over soft constraints. The soft constraints (S_1 – S_5) are rank four, before the distance constraint (D) which is the less important objective.

$$\text{minimize}_{LEX} (f_H, \sum f_{R_i}, f_P, \sum f_{S_i}, f_D) \quad (23.1)$$

The use of a lexicographic order between different sets of constraints is justified by its simplicity and the fact that it is easily understandable by users. Unlike the INRC2010 problem model, it is not necessary to specify weights for each constraint. Thus, in the proposed model, one occurrence of an unsatisfied constraint corresponds to a cost of one unit in the related rank. The lexicographic order, which may appear complicated in the objective function, is in fact accessible for users without requiring particular knowledge in optimization.

23.2.3 Optimization Procedures

The procedure used for the reoptimization of solutions is the same as the procedure that provides the initial solutions. It is an Iterated Local-Search (ILS) [9] which basically alternates between an improvement phase and a perturbation step for exploring the solution space with one solution. In this section we provide a general description of the optimization procedure and we refer the reader to [10] for additional details on the implemented ILS procedure.

ILS is a trajectory metaheuristic, which means that the exploration is made with one solution that “moves” in the search space. One iteration consists of, a perturbation step applied on the current solution to diversify the search and escape local-optima, then an improvement of the perturbed solution by local-search, and finally the application of an acceptance criterion to determine if the next iteration starts from the newly improved solution or from the last accepted solution. In the implemented ILS, the improvement phase is a Variable Neighborhood Descent (VND) [6]. The neighborhood structures used for the VND are based on block-swap moves. As illustrated in the left hand side of Fig. 23.1, a move is an exchange between two employees of a block of consecutive assignments. The size of the blocks varies between 1 and 7 during the VND. The perturbation step applies random moves to the solution using a different neighborhood structure to that of VND. A perturbation

move is illustrated in the right hand side of Fig. 23.1. The moves for the perturbation are rotations of assignments' blocks between three employees. The advantage of such a rotation between three employees is that a perturbation move cannot be easily undone using block-swap moves. Concerning the acceptance criterion, a simple rule that only accepts improving solution has been adopted.

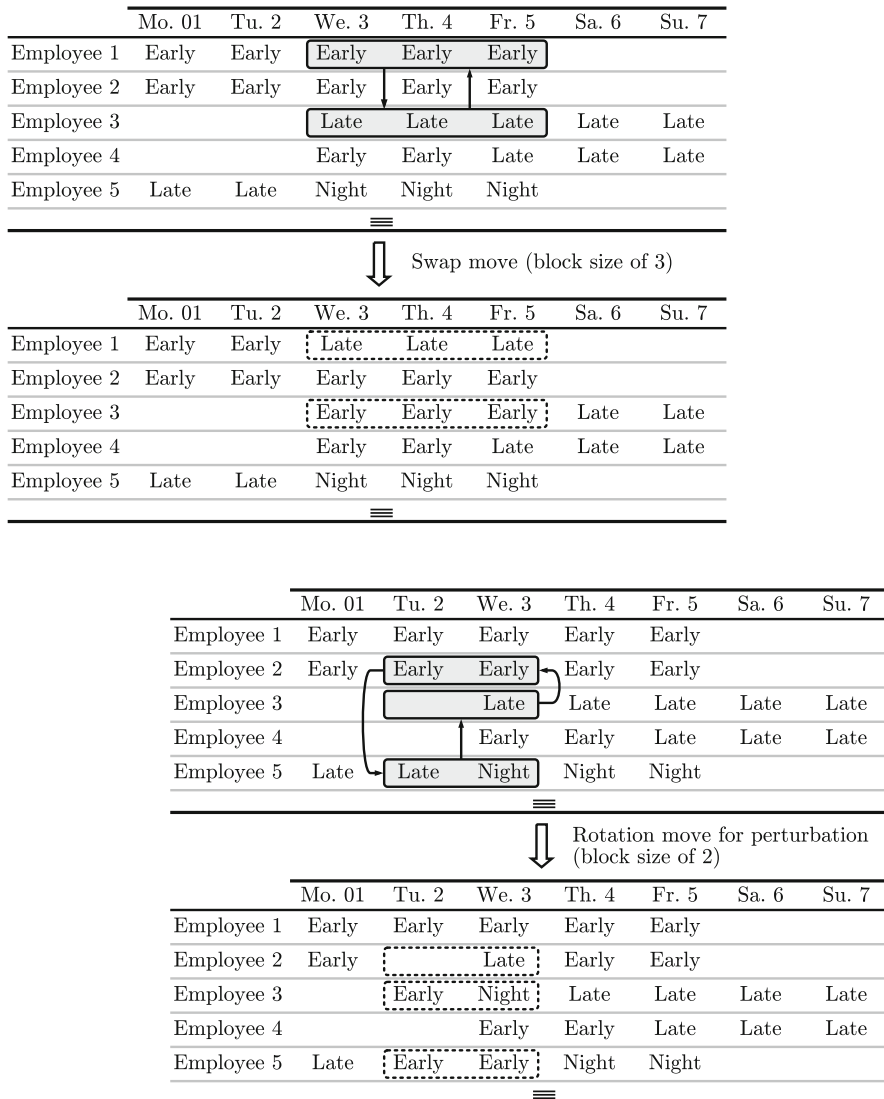


Fig. 23.1 Illustration of moves used in the ILS procedure. The first move (first two schedules at the top of the figure) illustrates the block-swap moves used in the VND procedure. The second move (third and fourth schedules) shows a rotation move as used in the perturbation procedure

For the reoptimization, the performance of the procedure is a key element. As we mentioned previously, the reoptimization should last only a few seconds to be appropriate for an interactive context. For the experiment the duration of the reoptimization is fixed to 5 s. In order to meet this requirement, the implemented ILS procedure makes extensive use of delta-evaluation of solutions, and also exploits high-level parallelism. The parallel version of ILS is obtained by running multiple concurrent ILS procedures that exchange their best found solutions. This simple parallel strategy only aims at speeding up computation by using multiple threads. On the INRC2010 benchmark the results of the implemented ILS procedure are comparable to the results of state of the art metaheuristics [10].

23.3 Experiment

23.3.1 Objectives

The proposed experiment aims at evaluating the interactive reoptimization procedure presented in the previous section with real user interactions. For this evaluation, the results obtained by interactive reoptimization are compared to results of a manual adjustment of solutions, and also compared with ideal solutions which require no adjustments. This comparison between reoptimized solutions, manually adjusted solutions and ideal solutions should provide a measure of the cost gain obtained by the interactive reoptimization. In addition, the experiment should assess the impact of manual adjustment of solutions when it is not assisted by a reoptimization procedure. Finally, the comparison between reoptimized and ideal solutions should provide information on the limits of the proposed interactive reoptimization approach.

In a real decision context it would be difficult to evaluate quantitatively the interactive reoptimization approach. The inaccuracies of the optimization model that necessitate to adjust solutions are generally not clearly specified (otherwise these inaccuracies would be resolved in the optimization model). Therefore, it is difficult to determine ideal solutions, and the comparison between interactive reoptimization and manual adjustment would be delicate without knowing exactly if the compared solutions satisfy the same criteria. These obstacles are overcome in the experiment by controlling which aspects of the solutions need to be adjusted. More precisely, during the experiment, the users are asked to modify solutions according to given constraints. Thus, it is possible to determine if the expressed constraints are successfully integrated, and it becomes meaningful to compare solutions obtained by interactive reoptimization with manually adjusted solutions and ideal solutions.

The task analyzed during the experiment is the modification of an *initial roster* to satisfy a given constraint, called *missing constraint*. Subjects completed this task with different missing constraints and initial rosters, each representing a *scenario*. For instance, one of the scenarios consists in modifying a roster so that a particular employee has no shift assigned on Wednesdays. The scenarios are completed

by subjects with two different methods, namely manual edition and interactive re-optimization. For both approaches, the Graphical User Interface (GUI) is the same (see Fig. 23.2) but the means of interaction are limited to either manual adjustment actions or the use of interactive reoptimization tools. In order to compare the results of the two modes of interaction, subjects are asked to satisfy the missing constraints with the following priorities: First, the modifications must not impact work regulations constraints, then the missing constraint must be satisfied, and finally the number of unsatisfied soft constraint should be minimized. Thus, for scenarios where the missing constraint is satisfied, solutions can be compared on the basis of the cost of soft constraints.

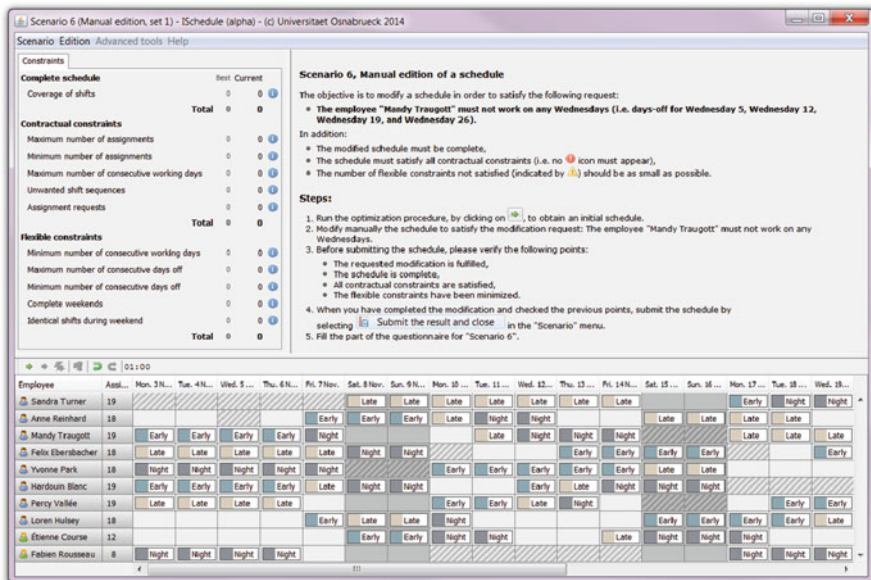


Fig. 23.2 Graphical user interface of the software implemented for the experiment

23.3.2 Method

23.3.2.1 Subjects and Procedures

The subjects for the experiment are voluntary undergraduate and graduate students of the University of Osnabruck. Sixteen subjects took part in the experiment. The course of the experiment is summarized in Fig. 23.3. Each subject completed a session on manual edition and another session on interactive reoptimization. To minimize possible order effects, a group of subjects started with the interactive reopti-

mization task and the remaining subjects started with the manual edition task. For the two sessions, the task was explained to the subjects and two training scenarios were completed and verified by the experimenter before starting the evaluated scenarios. Then, the subjects completed ten scenarios. For each scenario, the subject had 10 min for adjusting a provided roster according to a missing constraint. Then, the subject filled out a short questionnaire in which it is asked if the task has been understood and also how the subject evaluate his/her own performance.

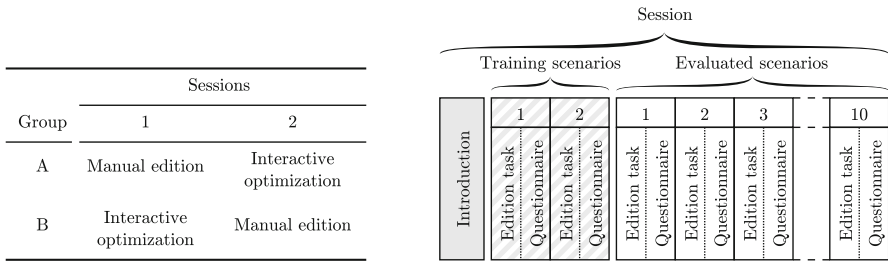


Fig. 23.3 Course of the experiment for the two groups. Each session has the same structure (*on the right*)

23.3.2.2 Details of Scenarios

The ten scenarios that are completed during the manual edition task are the same as the scenarios completed using interactive reoptimization. The list of the respective missing constraints is given in Table 23.1. To simplify the task for the subjects, all scenarios are based on similar shift scheduling problems with small variations in day off requests and in initial rosters. However, it should be noted that scenarios completed by a user are paired between the manual edition session and the interactive reoptimization session (i.e. for a scenario completed by a subject with manual edition the exact same problem and initial roster is used for the related scenario in interactive reoptimization). The variations in problems and in initial rosters should ensure that average results per scenario are not altered by specific configurations of initial rosters.

Since the subjects are not experts in scheduling, the problems have been simplified for being tractable without prior knowledge in shift scheduling. Only ten employees are considered, with three possible shifts (*Early*, *Late* and *Night*) and two types of contract (*Full-time* and *Part-time*). Also for simplifying the task, the initial rosters that are optimal pre-computed rosters satisfy all work regulations and only one or no soft constraint is unsatisfied. Thus, subjects do not have to put much effort on the satisfaction of work regulation constraints and can essentially focus on the satisfaction of the given missing constraint and the minimization of unsatisfied soft constraints.

Table 23.1 List of the missing constraints that had to be integrated by subjects in initial rosters

Scenario	Missing constraint
1	The employee $E1^a$ must not work on the first weekend
2	The employees $E2$ and $E3$ must work only on Early shifts for the whole planning period
3	The employee $E4$ must not work on $D1$ and $D2$
4	The employee $E5$ must not work the second and fourth weekends
5	The employee $E6$ must work five consecutive days from $D3$
6	The employee $E7$ must not work on any Wednesdays
7	Part-time employees must not work on Night shifts
8	For part-time employees, the Monday must be free after a working Sunday
9	For all employees, no Night shift must be assigned on Friday when the weekend is free
10	For all employees, a Night shift must not be assigned the day after an Early shift

^aFor the experiment, the codes for employees ($E\#$) have been replaced by generated names, and days ($D\#$) by real dates

23.3.2.3 Interactions

As previously mentioned, the same GUI presented in Fig. 23.2 is used for both the manual edition task and the interactive reoptimization task. It is composed of three panels. The upper right panel displays the instructions. When the subject loads a scenario, this panel indicates the missing constraint to introduce and recalls the steps for modifying the initial roster. The upper left panel gives a summary of the constraints unsatisfied for the current roster. The subject can also obtain a description of the constraints and their parameters from this panel. When the roster is modified, the values for unsatisfied constraints are instantaneously updated. Finally, the lower panel display the current roster. All unsatisfied constraints are represented directly on the affected assignments and the subject can obtain a precise description of them by selecting the assignments. When a modification of the roster is made, by manual edition or reoptimization, the roster and all indications concerning unsatisfied constraints are instantaneously updated. In addition, the last modifications made on the roster are highlighted to easily keep track of the changes. For modifying the roster, the subject directly interacts with the assignments of the roster. The actions available for the manual edition task and the reoptimization task are as follows.

For manually editing a roster, the subject can swap assignments by drag-and-drop between any employee. In addition, assignments can be removed and then reassigned to any employee using drag-and-drop. The list of shifts for which the demand is not fulfilled appears above the corresponding days. Any modification made on the roster can be undone and redone using buttons in the toolbar of the roster.

For the interactive reoptimization task, only the reoptimization tools are enabled and the roster cannot be directly modified using drag-and-drop of assignments. The subject can set assignment preferences using a contextual menu. A right mouse-click

on an assignment opens a menu in which the subject can select the preferred and unwanted assignments (that includes the possibility to set a preference for a day-off). In addition, it is possible to change or clear assignment preferences at any time. The assignments for which a preference has been defined are indicated on the roster. To introduce the desired changes (i.e. reflect assignment preferences) the subject clicks on a reoptimization button in the toolbar of the roster. The roster is then reoptimized for 5 s and the changes are dynamically displayed. As for the manual edition, the user can undo and redo any change, including the definition of assignment preferences and the reoptimization, using buttons in the toolbar of the roster.

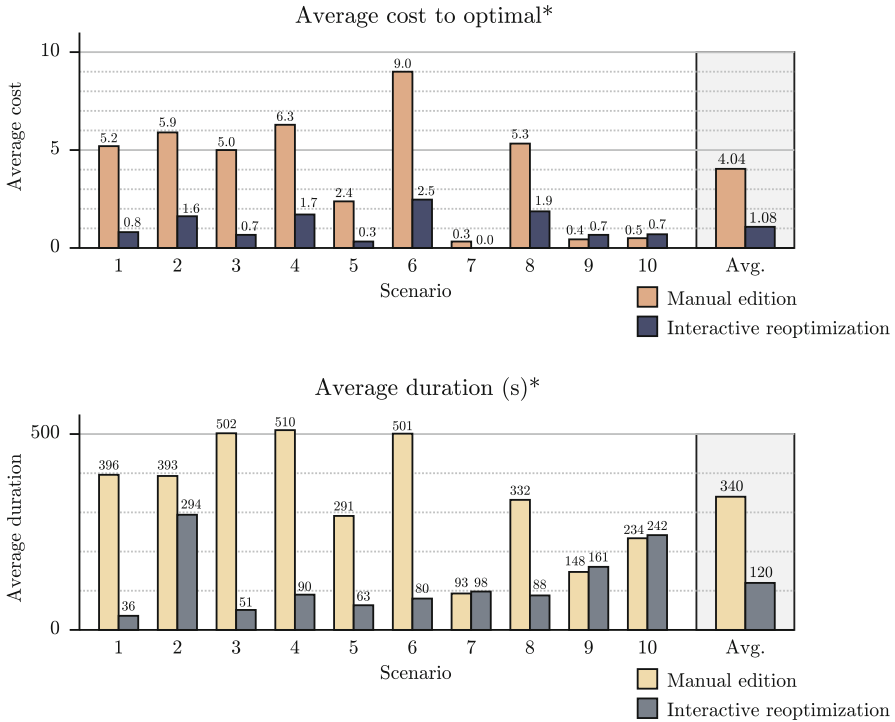
23.4 Results

During the experiment, the rosters obtained by subjects using manual edition and interactive reoptimization are compared to ideal solutions. These ideal solutions correspond to rosters that optimally express the missing constraints (i.e. satisfy the work regulations, the missing constraint and have the minimum possible number of unsatisfied soft constraints). Ideal solutions are computed for each scenario using the ILS procedure applied on a model that contains the missing constraint. Optimality has been verified using global lower-bound values.

For the results obtained by the subjects, only the solutions that satisfy the work regulations and the missing constraints are considered. When the missing constraint is not satisfied in a roster adjusted by a subject, it is not possible to determine if it is due to a misunderstanding of the subject, the result of inattention, or related to the difficulty of the task. Thus, we removed those results to only compare successful adjustments of rosters.

Since all of the compared solutions satisfy the work regulations and the missing constraints, the *cost* of a solution is defined as the number of soft constraints unsatisfied. The charts in Fig. 23.4 reports the average cost difference between solutions obtained by subjects and ideal solutions. In addition, the average duration of the adjustment process is reported. This duration correspond to the time from the first interaction with the initial roster to the end of the last action performed by the subject. For the interactive reoptimization task, this duration includes the time taken by the reoptimization procedure.

The results in Fig. 23.4 confirm that the interactive reoptimization approach is globally more efficient than manual edition for adjusting solutions. But more importantly, these results provide a quantitative evaluation of the impact of manual adjustment of solutions. The manual edition of solutions introduces on average 4.04 unsatisfied soft constraints where the interactive optimization produces on average 1.08 unsatisfied soft constraint. The worst scenario for both manual edition and interactive reoptimization is the scenario 6 for which the average cost differences to optimal values are respectively 9 for manual edition and 2.47 for interactive reoptimization. For scenarios 1 and 3, where only two assignments have to be changed in the initial roster, the manual edition introduces respectively 5.2 and 5 unsatisfied



* Excluding results for which the missing constraints has not been successfully introduced in the schedule.

Fig. 23.4 Average results for the manual edition task and the interactive reoptimization task

constraints. These results show the substantial impact of manual adjustment of solutions on their quality. In addition, these comparative results illustrate the gains in terms of cost and duration achieved by the interactive reoptimization approach.

A closer look at the results reveals few cases where the interactive reoptimization approach provides worse results than the manual edition setting. For scenarios 9 and 10, the average costs of solutions obtained by manual edition are better than the average costs achieved by interactive reoptimization. It should be noted, however, that these average costs are below 1 for both manual edition and interactive reoptimization. Regarding the average durations, the average times for completing scenarios 7, 9 and 10 are slightly better for manual edition than for interactive reoptimization. These cases where the interactive reoptimization approach has similar or inferior results than the manual edition setting does not question the overall effectiveness of the interactive reoptimization but expose some limits of the implemented approach. These limits are summarized below.

Computational Performance of ILS The reoptimization of rosters is a challenging optimization problem, in particular with the limited computation time. Metaheuristics such as ILS appear to be appropriate for reoptimization thanks to their capacity to provide good solutions in a reasonable time. However, there is some room for improving the implemented ILS and obtaining better solutions.

Expressiveness of Preferences In the proposed interactive reoptimization method, the user can only define preferences related to single assignments. When, in scenarios 9 and 10, it is asked to adjust a roster according to preferred or unwanted sequences of shifts, the preferences on assignments become less efficient. In fact for these two scenarios, when a preference is set for adjusting an inadequate sequence of shifts, the reoptimization affects the rest of the solution and may reintroduce the inadequate sequence at another place of the roster. In this case, it is necessary to proceed with multiple iterations of preference definition and reoptimization, although the distance constraint tends to reduce the number of iterations. This could result in the definition of assignment preferences that overconstrain the problem (e.g. the initial assignment preferences may no longer be necessary after several iterations) and thus impair the quality of solutions. The design of additional tools for defining preferences seems to be necessary to address this problem.

Interactions Finally, it should be noted that for the interactive reoptimization task the subjects had no means to manually adjust the roster, and in some cases the visualization of the roster allows the subjects to identify efficient moves that are hardly made by the ILS. The good performances on the manual edition task for scenarios 7, 9 and 10 shows that in particular cases it could be interesting to exploit user heuristics for improving solutions. A possible approach would be to combine both manual and interactive approaches for adjusting solution, but it raises the problem that a user may be reluctant to use the interactive reoptimization approach. In this direction, it seems necessary to study the usability and acceptance of methods that combines multiple interaction mechanisms.

23.5 Conclusion and Perspectives

In this paper, we proposed an interactive reoptimization method for adjusting solutions when some inaccuracies in an optimization model need to be solved by the user of the optimization system. This interactive process is studied for a shift scheduling problem. The method proposed for reoptimizing solutions and integrating changes requested by a user is an Iterated Local Search (ILS) procedure. We proposed an experiment to evaluate this interactive reoptimization approach. The experiment was conducted with 16 subjects and ten different scenarios. The results of the interactive reoptimization approach were compared with solutions obtained by manual adjustment and with ideal solutions. This comparison shows the value of a global optimization method such as the ILS procedure for integrating efficiently some changes in a solution. In addition, the results of the experiment demonstrate the impact of manual adjustment of solutions on their quality. Finally, this experiment revealed some limits of the implemented interactive reoptimization method. The perspectives of this work are directly connected to the observations made on the results of the experiment. Further works will concern the improvement of performance of the reoptimization procedure, the investigation of additional interaction means, and also will address usability and acceptance of the proposed interactive approach.

Acknowledgements This work was supported by the Deutsche Forschungsgemeinschaft (DFG), under grant ME 4045/2-1, for the project “Interactive metaheuristics for optimization-based decision support systems”. I acknowledge the support of Google, through the Google Focused Grant Program on “Mathematical Optimization and Combinatorial Optimization in Europe” (2012), which allowed us to initiate this study. I want to thank Sigrid Knust for her support throughout the project.

References

1. D. Anderson, E. Anderson, N. Lesh, J. Marks, B. Mirtich, D. Ratajczak, K. Ryall, Human-guided simple search, in *AAAI 2000*, pp. 209–216 (2000)
2. G. Ausiello, V. Bonifaci, B. Escoffier, Complexity and approximation in reoptimization, in *Computability in Context: Computation and Logic in the Real World* (Imperial College Press, London, 2007), pp. 101–129. ISBN:978-1-84816-245-7
3. A.T. Ernst, H. Jiang, M. Krishnamoorthy, D. Sier, Staff scheduling and rostering: a review of applications, methods and models. *Eur. J. Oper. Res.* **153**(1), 3–27 (2004)
4. G.A. Forgyon, An architecture for the integration of decision making support functionalities, in *Decision Making Support Systems: Achievements and Challenges for the New Decade*, Idea Group Publishing, Hershey, 2002, pp. 1–19
5. S. Hamel, J. Gaudreault, C.-G. Quimper, M. Bouchard, P. Marier, Human-machine interaction for real-time linear optimization, in *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 673–680 (2012)
6. P. Hansen, N. Mladenović, Variable neighborhood search, in *Handbook of Metaheuristics* (Kluwer Academic, Boston, 2003), pp. 145–184
7. S. Haspeslagh, P.D. Causmaecker, A. Schaerf, M. Stølevik, The first international nurse rostering competition 2010. *Ann. Oper. Res.* **218**(1), 221–236 (2014)
8. G.W. Klau, N. Lesh, J. Marks, M. Mitzenmacher, Human-guided search. *J. Heuristics* **16**(3), 289–310 (2010)
9. H.R. Lourenço, O.C. Martin, T. Stützle, Iterated local search: framework and applications, in *Handbook of Metaheuristics* (Springer, Berlin, 2010), pp. 363–397
10. D. Meignan, A heuristic approach to schedule reoptimization in the context of interactive optimization, in *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation* (ACM, New York, 2014), pp. 461–468
11. D. Meignan, An experimental investigation of reoptimization for shift scheduling, in *Proceedings of the 11th Metaheuristics International Conference (MIC'15)* (2015)
12. D. Meignan, S. Knust, J.-M. Frayret, G. Pesant, N. Gaud, A review and taxonomy of interactive optimization methods in operations research. *ACM Trans. Interactive Intell. Syst.* **5**(3), 17 (2015)
13. M.L. Pinedo, Design and implementation of scheduling systems: basic concepts, in *Scheduling Theory, Algorithms, and Systems*, 4th edn. (Springer, Berlin, 2012), pp. 459–483
14. H. Shachnai, G. Tamir, T. Tamir, A theory and algorithms for combinatorial reoptimization, in *LATIN 2012: Theoretical Informatics*. Lecture Notes in Computer Science, vol. 7256 (Springer, Berlin, 2012), pp. 618–630
15. J. Shim, M. Warkentin, J.F. Courtney, D.J. Power, R. Sharda, C. Carlsson, Past, present, and future of decision support technology. *Decis. Support. Syst.* **33**(2), 111–126 (2002)
16. A. Zych, Reoptimization of NP-hard problems. PhD thesis, Eidgenössische Technische Hochschule ETH Zürich (2012). Nr. 20257

Chapter 24

Surrogate-Assisted Multiobjective Evolutionary Algorithm for Fuzzy Job Shop Problems

Juan José Palacios, Jorge Puente, Camino R. Vela, Inés González-Rodríguez, and El-Ghazali Talbi

Abstract We consider a job shop scheduling problem with uncertain processing times modelled as triangular fuzzy numbers and propose a multiobjective surrogate-assisted evolutionary algorithm to optimise not only the schedule's fuzzy makespan but also the robustness of schedules with respect to different perturbations in the durations. The surrogate model is defined to avoid evaluating the robustness measure for some individuals and estimate it instead based on the robustness values of neighbouring individuals, where neighbour proximity is evaluated based on the similarity of fuzzy makespan values. The experimental results show that by using fitness estimation, it is possible to reach good fitness levels much faster than if all individuals are evaluated.

Keywords Fuzzy job shop • Robust scheduling • Multiobjective evolutionary algorithm • Surrogate fitness

24.1 Introduction

Scheduling problems form an important body of research since the late fifties with multiple applications in industry, finances, welfare, etc. Traditionally, scheduling has been treated as a deterministic problem that assumes precise knowledge of all data. However, modelling real-world problems usually involves processing uncertainty and flexibility. In the literature we find different proposals for dealing with uncertainty in scheduling [14], either finding solutions which adapt dynamically

J.J. Palacios • J. Puente • C.R. Vela

Department of Computing, University of Oviedo, Gijón, Spain

e-mail: palaciosjuan@uniovi.es; puente@uniovi.es; crvela@uniovi.es

I. González-Rodríguez (✉)

Department of Mathematics, Statistics and Computing, University of Cantabria, Santander, Spain

e-mail: gonzalezri@unican.es

E.-G. Talbi

INRIA Laboratory, CRISTAL/CNRS, University Lille 1, Villeneuve d'Ascq, France

e-mail: el-ghazali.talbi@univ-lille1.fr

to changes or incorporating available knowledge about possible changes to the solution. In particular, fuzzy sets have contributed to bridge the gap between classical techniques and real-world user needs, serving both for handling flexible constraints and uncertain data [26]. They are also emerging as an interesting tool for improving solution robustness, a much-desired property in real-life applications [18].

When the improvement in robustness must not be obtained at the cost of losing performance quality in the solutions, we face a bi-objective scheduling problem. In general there is a growing interest in multiobjective optimisation for scheduling and, given its complexity, in the use of metaheuristic techniques to solve these problems, as shown in [7] among others. Specifically, the multiobjective fuzzy job shop problem is receiving an increasing attention, mostly to optimise objective functions related to makespan and due-date satisfaction. Existing proposals include genetic algorithms [12], differential evolution algorithms [15], or hybrid strategies like the genetic simulated-annealing algorithm from [25]. Interestingly, the latter contemplates finding both robust and satisfactory schedules, although the robustness optimisation criterion is based on the worst-case approach which can be too conservative in cases where the worst case is not that critical and instead an overall acceptable performance might be more adequate.

Roughly speaking, a schedule is said to be *robust* if it minimises the effect of executional uncertainties on its primary performance measure [1], the makespan in our case, so the robustness of one schedule can be only measured after executing it in a real environment. In absence of real execution, we can use Monte-Carlo simulations to approximate the robustness value of every schedule, but even with this approximation the number of simulations required to compute this value translate into an excessive computational cost for a fitness function in a evolutionary algorithm. This suggests resorting to surrogate-assisted evolutionary computation, which was mainly motivated to reduce computational time in problems where complex simulations are involved [16].

The idea of approximating fitness values of some individuals based on information generated during the run, i.e. based on fitness values of individuals generated previously, has lately gained increasing attention [16]. In the simplest case, the fitness of a new individual is derived from its parents' fitnesses. Other approaches attempt to construct a more global model of the fitness landscape based on previous evaluations. Several such approaches can be found in the literature, mainly differing in the model that is used to approximate the landscape and the selection of data points used to construct the model [5]. The idea is to keep previous evaluations in a history and select the closest neighbours to build a specific estimation model.

In the following, we will consider the bi-objective fuzzy job shop problem with the goal of optimising both makespan and robustness. We will propose to solve it with a multiobjective evolutionary algorithm (MOEA) where the fitness related to robustness is found via surrogates, considering closest neighbours in terms of approximation of fuzzy values, using the degree of similarity between fuzzy sets to have a proximity measure.

The rest of the paper is organised as follows. Section 24.2 introduces the fuzzy job shop scheduling problem and a related robustness measure. Section 24.3 describes the multiobjective evolutionary algorithm proposed to solve the problem, including the surrogate model for evaluating the robustness fitness function. We then present and analyse some experimental results in Sect. 24.4 and finish with some conclusions and future work in Sect. 24.5.

24.2 Job Shop Scheduling with Uncertain Durations

The *job shop scheduling problem*, also denoted *JSP*, consists in scheduling a set of jobs $\{J_1, \dots, J_n\}$ on a set of physical resources or machines $\{M_1, \dots, M_m\}$, subject to a set of constraints. There are *precedence constraints*, so each job J_i , $i = 1, \dots, n$, consists of m tasks $\{\theta_{i1}, \dots, \theta_{im}\}$ to be sequentially scheduled. Also, there are *capacity constraints*, whereby each task θ_{ij} requires the uninterrupted and exclusive use of one of the machines for its whole processing time. A solution to this problem is a schedule (an allocation of starting times for all tasks) which, besides being *feasible*, in the sense that precedence and capacity constraints hold, is optimal according to some criteria, for instance, that the makespan is minimal or its robustness is maximal.

24.2.1 Uncertain Durations

In real-life applications, it is often the case that the exact duration of a task, i.e. the time it takes to be processed, is not known in advance, and only some uncertain knowledge is available. Such knowledge can be modelled using a *triangular fuzzy number* or TFN, given by an interval $[a^1, a^3]$ of possible values and a modal value a^2 in it. For a TFN A , denoted $A = (a^1, a^2, a^3)$, the membership function takes the following triangular shape:

$$\mu_A(x) = \begin{cases} \frac{x-a^1}{a^2-a^1} & : a^1 \leq x \leq a^2 \\ \frac{x-a^3}{a^2-a^3} & : a^2 < x \leq a^3 \\ 0 & : x < a^1 \text{ or } a^3 < x \end{cases} \quad (24.1)$$

In the job shop, we essentially need two operations on fuzzy numbers, the sum and the maximum. These are obtained by extending the corresponding operations on real numbers using the *Extension Principle*. However, computing the resulting expression is cumbersome, if not intractable. For the sake of simplicity and tractability of numerical calculations, we follow [11] and approximate the results of these operations, evaluating the operation only on the three defining points of each TFN. It turns out that for any pair of TFNs A and B , the approximated sum $A + B \approx$

$(a^1 + b^1, a^2 + b^2, a^3 + b^3)$ coincides with the actual sum of TFNs; this may not be the case for the maximum $\max(A, B) \approx (\max(a^1, b^1), \max(a^2, b^2), \max(a^3, b^3))$, although they have identical support and modal value.

The membership function of a fuzzy number can be interpreted as a possibility distribution on the real numbers. This allows to define its expected value, given for a TFN A by $E[A] = \frac{1}{4}(a^1 + 2a^2 + a^3)$. It coincides, among others, with the neutral scalar substitute of a fuzzy interval and the centre of gravity of its mean value [9]. It induces a total ordering \leq_E in the set of fuzzy numbers, where for any two fuzzy numbers A, B $A \leq_E B$ if and only if $E[A] \leq E[B]$.

24.2.2 Robust Scheduling

A fuzzy schedule does not provide exact starting times for each task. Instead, it gives a fuzzy interval of possible values for each starting time, provided that tasks are executed in the order determined by the schedule. In fact, it is impossible to predict what the exact time-schedule will be, because it depends on the realisation of the task's durations, which is not known yet. This idea is the basis for a semantics for fuzzy schedules from [13] by which solutions to the fuzzy job shop should be understood as a-priori solutions, also called baseline or predictive schedules in the literature [14]. When tasks are executed according to the ordering provided by the fuzzy schedule we shall know their real duration and, hence, obtain a real (executed) schedule, the a-posteriori solution with deterministic times. Clearly, it is desirable that a fuzzy solution yields reasonably good executed schedules at the moment of its practical use, in clear relation with the concept of schedule robustness.

As already mentioned, we consider that a schedule is *robust* if it minimises the effect on the makespan of executional uncertainties. This straightforward definition may, however, be subject to many different interpretations when it comes to specifying robustness measures [22]. In this work, we will consider only uncertainties in task processing times and we shall adopt the concept of ε -robustness proposed in [3] for stochastic scheduling, already adapted to the fuzzy flexible job shop in [20]. This definition states that a predictive schedule is considered to be robust if the quality of the eventually executed schedule is close to the quality of the predictive schedule. In particular, for the fuzzy job shop, a predictive schedule with makespan value $C_{max,pred}$ (a TFN) is ε -robust for a given ε if the objective value $C_{max,ex}$ of the eventually executed schedule (a real value) is such that:

$$(1 - \varepsilon) \leq \frac{C_{max,ex}}{E[C_{max,pred}]} \leq (1 + \varepsilon) \quad (24.2)$$

or, equivalently,

$$\frac{|C_{max,ex} - E[C_{max,pred}]|}{E[C_{max,pred}]} \leq \varepsilon. \quad (24.3)$$

That is, the relative error of the estimation made by the predictive schedule (i.e. its expected makespan) is bounded by ε . Obviously, the smaller ε is, the better.

Notice however that this definition requires a real execution of the problem which may not always be available. Consider for instance the synthetic problems commonly used in the literature as benchmarks. In this case, following [20], we provide an approximation of the ε -robustness measure by means of a Monte-Carlo simulation. Given a fuzzy instance, we generate a sample of K possible realisations of that instance by assigning an exact duration to each task, that is K deterministic instances on which we can evaluate the ε -robustness of the solution. Now for each realisation $k = 1, \dots, K$, let $C_{max,k}$ denote the exact makespan obtained by executing tasks according to the ordering provided by a predictive schedule. Then, the *average ε -robustness* of the predictive schedule, denoted $\bar{\varepsilon}$, is calculated as:

$$\bar{\varepsilon} = \frac{1}{K} \sum_{k=1}^K \frac{|C_{max,k} - E[C_{max}]|}{E[C_{max}]}, \quad (24.4)$$

where $E[C_{max}]$ is the expected makespan estimated by the predictive schedule.

A crucial factor in this method is the way in which we sample deterministic durations for the tasks based on their fuzzy values. This is done by simulating exact durations for tasks following a probability distribution that is consistent with the possibility distribution μ_A defined by each fuzzy duration A . A simple approach consists in considering the uniform probability distribution that is bounded by the support of the TFN. This possibility-probability transformation is motivated by several results from the literature (see [10, 2]) that justify the use of TFNs as fuzzy counterparts to uniform probability distributions and model-free approximations of probability distributions with bounded support.

24.2.3 The Multiobjective Approach

In scheduling, when there is uncertainty in some of the input data, solution robustness becomes an important factor to be taken into account. Indeed, an optimal solution found for an ideal deterministic scenario (for instance, assuming that all durations take their modal value) may be of little or no use when it is executed if changes in the input data affect its real performance. Therefore, our aim in this work is to optimise both a performance or quality function, the expected makespan $E[C_{max}]$, as well as the robustness of the solution with respect to that function, the approximate measure $\bar{\varepsilon}$.

To optimise these two objective functions, we shall take a dominance-based approach. In general, for a minimisation problem with f_i , $i = 1, \dots, n$ objective functions, a solution s is said to be *dominated* by a solution s' , denoted $s' \succ s$ iff for each objective function f_i , $f_i(s') \leq f_i(s)$ and there exists at least one objective function

such that $f_i(s) < f_i(s)$. Our goal will then be to find non-dominated solutions to the FJSP with respect to $E[C_{max}]$ and $\bar{\varepsilon}$. To achieve this, we propose a dominance-based multiobjective evolutionary algorithm (MOEA) [23].

24.3 Multiobjective Evolutionary Algorithm

We propose a MOEA based on the well-known NSGA-II template [8]. An initial population is randomly created and evaluated and then the algorithm iterates over a number of generations, keeping a set of non-dominated solutions. At each iteration i , a new population $Off(P_i)$ is built from the current one P_i by applying the genetic operators of selection and recombination and then a replacement strategy is applied to obtain the next generation P_{i+1} . Finally, the stopping criterion can be at least one of the following: stop when no solution belonging to the set of non-dominated solutions is removed from this set after n_{iter} iterations or after a fixed number of iterations or after a given running time.

Solutions are encoded into chromosomes using permutations with repeated elements, which are permutations of the set of tasks, each being represented by its job number [4]. A given chromosome is decoded into the associated schedule, using an insertion strategy in the schedule-generating scheme [19]. This immediately gives the makespan expected value. The other fitness function is evaluated based on the degree of similarity between the fuzzy makespan of the current solution and a set of solutions, named *cache* hereafter, for which the $\bar{\varepsilon}$ value has been previously calculated. We shall refer to this algorithm as sMOEA.

24.3.1 The Surrogate Model

Evolutionary algorithms usually need a large number of fitness evaluations before obtaining a satisfying result. Either when an explicit fitness function does not exist or when the evaluation of the fitness is computationally very expensive, it becomes necessary to estimate the fitness value by an approximate model. This is indeed the case of one of our fitness functions, related to the objective of robustness.

Fitness approximation has been addressed from different areas, as can be seen for instance in [5, 16]. Techniques to manage surrogates for fitness evaluation include evaluating the fitness function only in some of the generations or in some individuals within a generation, having a pre-selection of offspring before evaluation, evaluating only those individuals that potentially have a good fitness value or choosing for re-evaluation representative individuals by clustering the population, to mention but a few.

Here we propose a new double approximation by using data sampling techniques. First, we run a Monte-Carlo simulation to provide a surrogate of the ε -robustness measure for the individuals in the cache, as explained in Sect. 24.2.2. Then we ap-

proximate the ε -robustness of a solution with that of the most similar solution in the set for which $\bar{\varepsilon}$ values have been previously computed, provided that this similarity exceeds a given threshold. The set of pre-evaluated solutions is named cache inspired in the cache memory of the computers. For this second part of the surrogate model there are several design decision that have to be made, namely, the similarity measure and the update strategy of the cache list. The similarity threshold and the size of the cache list are to be experimentally determined.

24.3.1.1 The Similarity Measure

Every decoded solution has a fuzzy makespan value, so we propose to use a similarity measure for TFNs to compare a given solution with every solution in the cache list based on fuzzy makespan values. The rationale behind this choice is that, on one hand, $\bar{\varepsilon}$ depends on the expected value of the fuzzy makespan and, on the other hand, the makespan of every deterministic realisation used to compute $\bar{\varepsilon}$ lies in the support of the fuzzy makespan.

In the literature we can find numerous proposals to quantify the degree of similarity between two fuzzy numbers using different descriptive parameters, such as the geometric distance, the perimeter, the area or the distance between the centres of gravity, etc. [6, 24]. However, most similarity functions are not adequate for our framework. In particular we need normalised similarity values in order to establish threshold values which are independent of the instance and this normalisation must be invariant to translations if they are to correspond to similarities in $\bar{\varepsilon}$ values. Additionally, similarity degrees must be easy to compute. That is, we look for simple but still representative measures.

In this paper, we consider a measure based on the so-called shared area between the fuzzy numbers. The shared area between fuzzy numbers with respect to the total area of these fuzzy numbers has been incorporated as a component of the measure of similarity of generalised fuzzy trapezoidal numbers in [24]. Here we are using TFNs, less complex than generalised fuzzy numbers, so we need only consider this value. The degree of similarity $S_{A,B}$ of A and B is then defined as

$$S_{A,B} = \frac{Area(A \cap B)}{Area(A \cup B)} \quad (24.5)$$

When $A \cap B$ is not a triangle, we approximate the area by the maximum triangle inscribed in this plane area. We will say that A and B are *approximately equal* given a small nonnegative number δ iff $S_{A,B} \leq \delta$.

24.3.1.2 The Update Strategy

The update strategy for the cache is motivated by the fact that, as the algorithm converges, the chance that a new solution lies in areas of the search space with bad

solutions becomes smaller. A new solution is thus added to the cache only if it is not similar enough to any of the elements already in the list, in the sense that they are not approximately equal as defined above. When this is the case, the solution is fully evaluated to obtain its \bar{e} value and added to the list. If this is full, the new solution replaces the one in the list that has not been used for longer to estimate the value of the robustness for another individual.

The cache is initially empty. Then, every individual in the initial population is processed to be inserted in the list following the replacement strategy above. Notice that by doing this the cache list, which has a prefixed size, may not be full once the whole initial population is analysed.

24.3.2 Genetic Operators

In the selection phase all chromosomes are randomly grouped into pairs, and then each of these pairs is mated to obtain two offspring. For the mating we have implemented one of the most extended crossover operators for the *JSP*, the Generalized Order Crossover (GOX). In order to preserve the diversity of individuals inside the population and prevent the algorithm from getting stuck in local optima, the insertion mutation strategy is also introduced.

The replacement strategy is a key factor in MOEA algorithms. Here we adopt a strategy based on the non-dominated sorting approach with diversity preservation from [8], that is, solutions belonging to a lower (better) non-domination rank are preferred and, between two solutions in the same non-dominance level, we prefer the solution located in the less crowded region. To introduce greater diversity in the algorithm, we remove from the pool of individuals those which are repeated, in the sense that there exists in the pool at least another individual having identical values for all objective functions. In the case that such elimination causes the pool to contain less individuals than the population size, all the non-repeated individuals pass onto the next generation, which is later completed with the best repeated individuals according to their rank level and crowding distance.

24.4 Experimental Study

For the experimental study, we consider one hard and well-known instance obtained by fuzzifying task durations of a crisp JSP classical benchmark, La29, as proposed in [21].

We start by trying to gain some insight into the effects of considering different similarity thresholds and cache sizes, respectively denoted δ and λ hereafter. We have generated a set of 1000 random solutions and, for different similarity thresholds δ ranging from 0.10 to 0.95, we have recorded the percentage of times (called hit rate) that a solution is approximately equal to at least one solution in a set of

size λ , with λ ranging from 10 to 200. We have seen that, for similarity thresholds under 0.80, the cache almost always contains an approximately equal solution for most of the λ values. Moreover, even when $\delta = 0.80$ and λ values over 100 are considered, we have hit rates near 100%. Translated into our surrogate algorithm this could mean that actual $\bar{\varepsilon}$ values will most likely be computed only for those solutions inserted in the cache in the first iterations. However, a different behaviour is observed for the same cache sizes ($\lambda \geq 100$) when $\delta = 0.90$ and $\delta = 0.95$ are considered. On the other hand, when these stricter similarity thresholds are considered with small λ values, there is little chance of finding an approximately equal solution in the cache, so most of the times a new solution will need to be completely evaluated, neutralising the potential effect of the surrogate approach in running times. That said, $\delta = 0.8$ offers more reasonable hit rates for small λ values, which support considering these smaller cache sizes in the experimental study.

We now proceed to consider different values for δ and λ and analyse their effects in running times and solution quality. For the sake of clarity, we will refer to the multiobjective algorithm using the surrogate model as sMOEA, while MOEA will refer to the algorithm that avoids surrogates and evaluates every new solution. As a result of a preliminary parametric analysis, the parameter setup is as follows: population size 100, crossover and mutation probabilities 1.0 and 0.1 respectively. Given the stochastic nature of the algorithm, it is run ten times, so ten different sets of non-dominated solutions are stored in order to obtain representative data.

In the literature we find many proposals to compare multiobjective algorithms. In this work we consider two metrics: the hypervolume, which is a quality indicator that combines both convergence and diversity measures and the unary additive ε -Indicator, which is a distance-based indicator [23]. This last indicator is calculated with respect to a reference set RF , which ideally should be the optimal Pareto front PO^* . However, since the benchmark instance used here has not been solved yet, this Pareto front is unknown. In consequence, we approximate it by the set of non-dominated elements of the union of all sets of solutions obtained so far in this experimentation [23]. Additionally, to avoid problems derived from the different scales of the objective functions, we normalise their values. More precisely, let $f_i^-(S) = \min\{f_i(s) : s \in S\}$ be a lower bound of the objective function f_i in the set S , and

$$f_i^+(S) = \max\{f_i(s) : s \in S\} + 0.05 * (\max\{f_i(s) : s \in S\} - \min\{f_i(s) : s \in S\})$$

an upper bound thereof, then the objective value $f_i(s)$ of each solution $s \in S$ is normalised as follows:

$$\forall i f_i(s) = \frac{f_i(s) - f_i^-(S)}{f_i^+(S) - f_i^-(S)}. \quad (24.6)$$

By taking this upper bound, we prevent the solutions from having a value equal to 1, which can be troublesome when computing the hypervolume. Indeed, solutions with objective values equal to 1 define a rectangle with null area, making them unsuitable for fair comparisons.

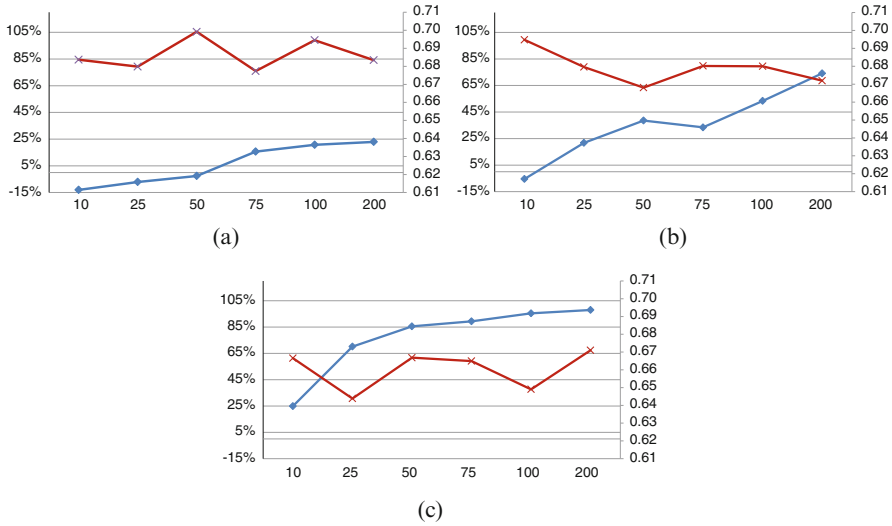


Fig. 24.1 Improvement and hypervolume values for different sizes of cache. *Red lines* represent hypervolumes and *blue lines* time reductions. (a) $\delta = 0.95$. (b) $\delta = 0.90$. (c) $\delta = 0.80$

In our experimental study, we first run the MOEA that evaluates $\bar{\epsilon}$ for every solution, using $ni_{iter} = 25$ non-improving consecutive iterations as stopping criterion. Besides the sets of non-dominated solutions, we register the number of generations needed for MOEA to converge (3950). Then, using this number of generations as stopping criterion, sMOEA is executed considering three δ values: 0.80, 0.90 and 0.95 and six λ values: 10, 25, 50, 75, 100 and 200. The sets of non-dominated solutions obtained after every run are then fully evaluated in order to obtain their actual $\bar{\epsilon}$ values that allow for fairer comparisons, so solutions that are dominated after the full evaluation are removed from the sets.

Figure 24.1 shows the improvement in running time of sMOEA w.r.t. MOEA and the hypervolume values for sMOEA given different combinations of λ and δ : Figure 24.1a for $\delta = 0.95$, Fig. 24.1b for $\delta = 0.90$ and Fig. 24.1c for $\delta = 0.80$. All figures show a primary Y-axis for time improvement in percentage (from -15% to 105%) and a secondary Y-axis for hypervolume values (from 0.61 to 0.71), being 0.6939 the hypervolume value reached by MOEA.

We can observe that, in general, increasing the similarity threshold improves the hypervolume and reduces the time improvement. This behavior is quite natural, as having a stricter threshold δ causes the algorithm to fully evaluate more solutions, thus having more accurate information, at the cost of having a longer runtime.

Notice as well that, even though time reductions are comparable, this is not the case for hypervolume (HV) values. When $\delta = 0.80$, the HV values obtained are in general much worse than those obtained with $\delta \geq 0.90$ (only with a cache size of 200 do they begin to be competitive). This leads us to reject 0.80 as an appropriate threshold for the similarity of solutions.

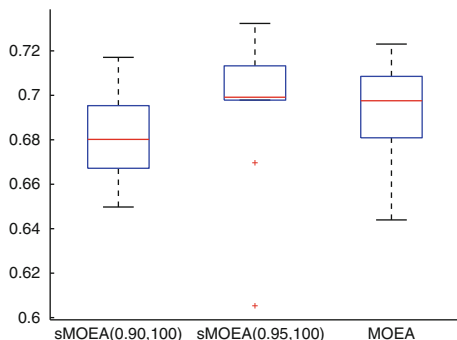


Fig. 24.2 HV for all runs of sMOEA(0.90,100), sMOEA(0.95,100) and MOEA (with no cache)

Focusing on Fig. 24.1a, b we can observe that, when the similarity threshold is high and the cache size is small, the processing time not only does not decrease, it even increases. Specifically, with $\delta = 0.95$ and $\lambda \leq 50$, and with $\delta = 0.90$ and $\lambda = 10$, CPU times of sMOEA are worse than those of MOEA. This is explained by the fact that it is unlikely to find an approximately equal individual in a small-sized cache, so most of individuals of the sMOEA are fully evaluated (as is the case with MOEA) and, additionally, sMOEA has to compare solutions and update the cache frequently, incurring in higher computational cost. For values such as $\delta = 0.95$ and $\lambda = 75$ or $\delta = 0.90$ and $\lambda = 50$, for which the cache is actually used by sMOEA, we can observe time improvements at some cost in HV values. More interestingly, when the cache size increases to 100, time reduction does not imply a loss in quality. We believe this is because the most similar solution in the cache is more likely to be really close to the current solution, having the effect of more reliability in the surrogates, leading to better HV values. This effect is nonetheless lost for $\lambda = 200$, probably because an excessively large cache causes the robustness fitness value to be estimated too often. Finally, we look at the ϵ -indicator for those sets of experiments where sMOEA does not consume more time resources than MOEA. We see that, only for $\lambda = 100$ and $\delta = 0.90$ or $\delta = 0.95$, the ϵ -indicator values obtained for sMOEA are similar to those of the MOEA (0.0712, 0.0791, and 0.0731 respectively).

Once a first filter has been applied to the different options for δ and λ values, we shall compare the two best variants of sMOEA (with $\lambda = 100$ and $\delta = 0.95$ or 0.90) and MOEA. Regarding runtime, sMOEA(0.90,100) obtains a greater reduction in CPU times (53% versus 21% of sMOEA(0.95,100)). In terms of quality, a first impression can be obtained from the box-plots in Fig. 24.2, which correspond to the HV values obtained using the three configurations. With the exception of two outsider values, HV values corresponding to the configuration sMOEA(0.95,100) appear to be slightly better than the rest. Having said this, differences in the box-plot are not big enough for a solid conclusion, making further comparisons necessary.

A more detailed comparison between the three algorithms, taking into account their multiobjective nature, can be done by means of the empirical attainment func-

tions (EAFs), which characterise the output of a stochastic multiobjective optimisation algorithm [17]. Figure 24.3 graphically plots the difference (using a gray scale) between the EAFs for algorithms MOEA (with no cache), sMOEA(0.95,100) and sMOEA(0.90,100), which allows us to identify the regions where one algorithm performs better than another. We can see that the solutions obtained with MOEA dominate those obtained by sMOEA(0.90,100) in almost every region with a low probability, slightly higher in the middle of the front, whereas solutions from sMOEA(0.90,100) almost never dominate those of MOEA with not null probability. The comparison between MOEA and sMOEA(0.95,100) reveals more equilibrium: the solutions from MOEA dominate with lower probability those of the surrogate version in some regions of the space, but they are dominated in other regions with a probability higher than 0. Finally, between both variants of the surrogate algorithm, sMOEA(0.95,100) is clearly better than sMOEA(0.90,100), since the difference between their EAFs is positive in almost every region and it is null when we make the opposite comparison.

Finally, a statistical analysis between HVs of the sets of solutions obtained with sMOEA(0.95,100) and MOEA is carried out, with a level of significance of 0.05 in every test considered. Once the normality of the samples and the homocedasticity are verified, a *t*-test is done with the result that there are no significant differences in terms of solution quality (*p*-value 0.918709). Notice that the lack of statistically significant differences, far from being a bad result, supports the interest of our proposal. Indeed, it indicates that the HV values obtained with the proposed sMOEA and MOEA are similar and therefore means that the use of surrogates does not have a bad influence in solution quality (in terms of hypervolume values), while it provides a reduction of over 20% in running time.

24.5 Conclusions

We have considered the job shop problem with uncertainty in the task durations modeled as fuzzy numbers. We have proposed to simultaneously optimise the makespan and the robustness of the solutions, understood as an overall acceptable performance under variations in the input data.

We have seen that a multiobjective evolutionary algorithm (MOEA) may produce good results for the FJSP, but at the same it is too time consuming. To overcome this drawback we have proposed a new surrogate model to spend less time in robustness evaluation. The resulting algorithm, termed sMOEA, has shown to be quite sensitive to the values of the parameters δ and λ . In particular, the combination of small δ values with large λ values gives rise to a fast method, at the cost of losing quality on solutions, while with large δ and small λ values the opposite happens. From a thorough experimental study, we have found a reasonable tradeoff between these parameters that allows sMOEA to reduce running times in more than 20% w.r.t. the original MOEA, without significant loss of solution quality in terms of hypervolumes.

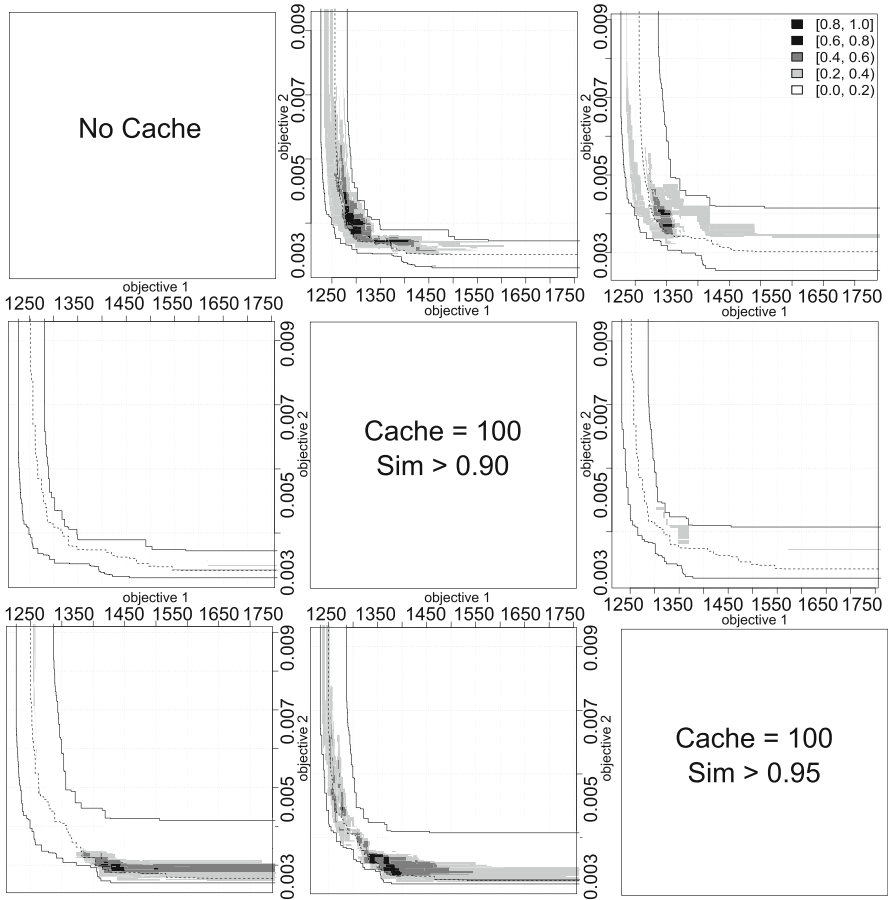


Fig. 24.3 Comparison between MOEA (no cache) and sMOEA(0.90/0.95,100) based on EAFs

Acknowledgements This research has been supported by the Spanish Government under Grant FEDER TIN2013-46511-C2-2-P.

References

1. H. Aytung, M.A. Lawley, K. McKay, M. Shantha, R. Uzsoy, Executing production schedules in the face of uncertainties: a review and some future directions. *Eur. J. Oper. Res.* **161**, 86–110 (2005)
2. C. Baudrit, D. Dubois, Practical representations of incomplete probabilistic knowledge. *Comput. Stat. Data Anal.* **51**, 86–108 (2006)
3. J. Bidot, T. Vidal, P. Laboire, A theoretic and practical framework for scheduling in stochastic environment. *J. Sched.* **12**, 315–344 (2009)

4. C. Bierwirth, A generalized permutation approach to jobshop scheduling with genetic algorithms. *OR Spectr.* **17**, 87–92 (1995)
5. J. Branke, C. Schmidt, Faster convergence by means of fitness estimation. *Soft Comput.* **9**, 13–20 (2005)
6. C.T. Chen, Extensions of the TOPSIS for group decision-making under fuzzy environment. *Fuzzy Set. Syst.* **114**, 1–9 (2000)
7. S. Dabia, E.G. Talbi, T. van Woensel, T. De Kok, Approximating multi-objective scheduling problems. *Comput. Oper. Res.* **40**, 1165–1175 (2013)
8. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
9. D. Dubois, H. Fargier, P. Fortemps, Fuzzy scheduling: modelling flexible constraints vs. coping with incomplete knowledge. *Eur. J. Oper. Res.* **147**, 231–252 (2003)
10. D. Dubois, L. Foulloy, G. Mauris, H. Prade, Probability-possibility transformations, triangular fuzzy sets and probabilistic inequalities. *Reliab. Comput.* **10**, 273–297 (2004)
11. P. Fortemps, Jobshop scheduling with imprecise durations: a fuzzy approach. *IEEE Trans. Fuzzy Syst.* **7**, 557–569 (1997)
12. O.A. Ghrayeb, A bi-criteria optimization: minimizing the integral value and spread of the fuzzy makespan of job shop scheduling problems. *Appl. Soft Comput.* **2**(3), 197–210 (2003)
13. I. González Rodríguez, J. Puente, C.R. Vela, R. Varela, Semantics of schedules for the fuzzy job shop problem. *IEEE Trans. Syst. Man Cybern. A* **38**(3), 655–666 (2008)
14. W. Herroelen, R. Leus, Project scheduling under uncertainty: survey and research potentials. *Eur. J. Oper. Res.* **165**, 289–306 (2005)
15. Y. Hu, M. Yin, X. Li, A novel objective function for job-shop scheduling problem with fuzzy processing time and fuzzy due date using differential evolution algorithm. *Int. J. Adv. Manuf. Technol.* **56**, 1125–1138 (2011)
16. Y. Jin, Surrogate-assisted evolutionary computation: recent advances and future challenges. *Swarm Evol. Comput.* **1**(2), 61–70 (2011)
17. M. Lopez-Ibañez, L. Paquete, T. Stützle, Exploratory analysis of stochastic local search algorithms in biobjective optimization, in *Experimental Methods for the Analysis of Optimization Algorithms*, chap. 9 (Springer, Berlin, 2010), pp. 209–222
18. J.J. Palacios, I. González-Rodríguez, C.R. Vela, J. Puente, Robust swarm optimisation for fuzzy open shop scheduling. *Nat. Comput.* **13**(2), 145–156 (2014)
19. J.J. Palacios, C.R. Vela, I. González-Rodríguez, J. Puente, Schedule generation schemes for job shop problems with fuzziness, in *Proceedings of ECAI 2014*, ed. by T. Schaub, G. Friedrich, B. O’Sullivan. *Frontiers in Artificial Intelligence and Applications*, vol. 263 (IOS Press, New York, 2014), pp. 687–692. doi:10.3233/978-1-61499-419-0-687
20. J.J. Palacios, I. González-Rodríguez, C.R. Vela, J. Puente, Coevolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop. *Fuzzy Set. Syst.* **278**, 81–97 (2015)
21. J. Puente, C.R. Vela, I. González-Rodríguez, Fast local search for fuzzy job shop scheduling, in *Proceedings of ECAI 2010* (IOS Press, New York, 2010), pp. 739–744
22. B. Roy, Robustness in operational research and decision aiding: a multi-faceted issue. *Eur. J. Oper. Res.* **200**, 629–638 (2010)
23. E.G. Talbi, *Metaheuristics. From Design to Implementation* (Wiley, New York, 2009)
24. E. Vicente, A. Mateos, A. Jiménez, A new similarity function for generalized trapezoidal fuzzy numbers, in *Artificial Intelligence and Soft Computing* (Springer, Berlin, 2013), pp. 400–411
25. B. Wang, Q. Li, X. Yang, X. Wang, Robust and satisfactory job shop scheduling under fuzzy processing times and flexible due dates, in *Proceedings of the 2010 IEEE International Conference on Automation and Logistics* (2010), pp. 575–580
26. B.K. Wong, V.S. Lai, A survey of the application of fuzzy set theory in production and operations management: 1998–2009. *Int. J. Prod. Econ.* **129**, 157–168 (2011)

Chapter 25

Towards a Novel Reidentification Method Using Metaheuristics

Tarik Ljouad, Aouatif Amine, Ayoub Al-Hamadi, and Mohammed Rziza

Abstract Tracking multiple moving objects in a video sequence can be formulated as a profile matching problem. Reidentifying a profile within a crowd is done by a matching process between the tracked person and the different moving individuals within the same frame. In that context, the feature matching task can be approximated to a search for the profile that maximizes a considered similarity measure. In this work, we introduce a novel Modified Cuckoo Search (MCS) based reidentification algorithm. A complex descriptor representing each moving person is built from different low level visual features such as the color and the texture components. We make use of a database that involves all previously detected descriptors, forming therefore a discrete search space where the sought solution is a descriptor and its quality is represented by its similarity to the query profile. The approach is evaluated within a multiple object tracking scenario, and a validation process using the normalized cross correlation method to accept or reject the obtained reidentification results is included. The experimental results show promising performances in terms of computation cost as well as reidentification rate.

Keywords Object tracking • Cuckoo search

T. Ljouad (✉)

Faculty of Sciences of Rabat (FSR), Mohammed V University, Rabat, Morocco

e-mail: ljouad.tarik@ieee.org

A. Amine

LGS Laboratory, National School of Applied Sciences (ENSA), Ibn Tofail University, Kenitra, Morocco

e-mail: amine_aouatif@univ-ibntofail.ac.ma

A. Al-Hamadi

Institute for Information Technology and Communications (IIKT),

Otto-von-Guericke-University Magdeburg, Magdeburg 39016, Germany

e-mail: ayoub.al-hamadi@ovgu.de

M. Rziza

Faculty of Sciences of Rabat (FSR), Mohammed V University, Rabat, Morocco

e-mail: rziza@fsr.ac.ma

25.1 Introduction

Many problems reported in the field of multiple object tracking, either in grayscale video or within an environment with similar parametric representation to that of the sought object, show that even with a sophisticated search strategy, the adopted object representation play a key role in the profile matching task. In fact, an object tracking algorithm's performance basically depend on two key factors: an intelligent browsing strategy to select a set of candidate regions with the highest probability to contain the sought object, and the use of a discriminant descriptor. After selecting the appropriate candidate set, the tracking task can be reduced to a matching task between the sought profile and the selected candidates descriptors, which is called a reidentification process (see Fig. 25.1).

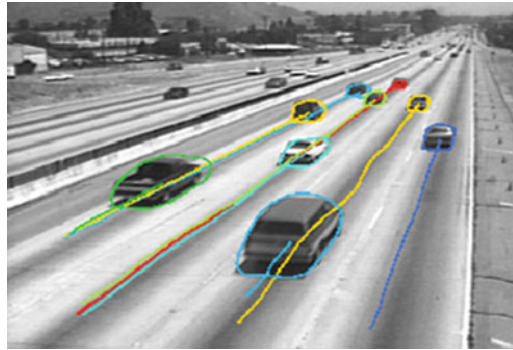


Fig. 25.1 Example of a reidentification task in a tracking scenario

In fact, the reidentification problem has been addressed in many researches because of the important role it plays in different application domains. The main aim of such an algorithm is to establish a reliable correspondence between different objects observed in cameras with non-overlapping fields of view by relying on their visual characteristics. Examples of application domains that makes use of such an algorithm could be the tracking devices across camera networks [18], multi-camera based event detection as well as pedestrian detection and extraction [13]. The challenge here is to associate different visual characteristics extracted from different videos in terms of image acquisition conditions, perspective, image resolution, poses, photometric calibration as well as crowded backgrounds.

The typical reidentification diagram is given in Fig. 25.2. Generally, the process should start with a detection phase in order to extract the location of all moving objects in the video, especially in case of a long video. Nevertheless, all existing approaches skip this phase and rely on the provided ground truth, assuming a perfectly accurate detection [6, 12]. The obtained locations are used to initialize the visual representation of the sought object. The extracted visual characteristics are supposed to be robust against affine transformation, as well as significant in case

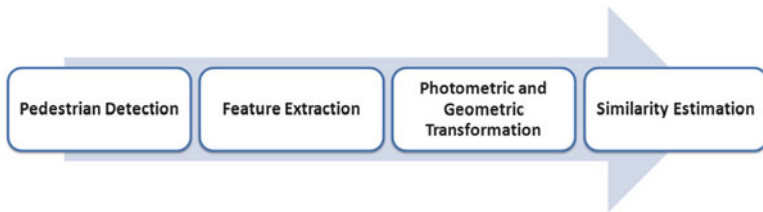


Fig. 25.2 Typical functional diagram of a reidentification process

of an occlusion. Different approaches in the state-of-the-art aim either at improving one or a combination of different modules of those exposed in the diagram (see Fig. 25.2).

The global characteristics describe the general aspect of the object’s color and texture [3, 23]. This information presents the major advantage of being invariant to the object’s alignment, pose variation as well as perspective changes. Nevertheless, their discriminative capability is relatively weak since they are unable to reproduce the spatial information of the descriptive itself. In order to tackle this weakness, many approaches adopted the patch concept during the matching process. Comparing two different images is then performed by evaluating the similarity between each patch in the reference image with its corresponding patch in the test image. The remaining question is then how to handle a misalignment issue. Although, it is possible to select a set of characteristics, and assign a corresponding weight to each one of them, this task might be corrupted by the initial choice of the extracted characteristics. The optimal solution would be a progressive training of selected feature from aligned data. Such a method is known as the deep learning technique [12, 20, 16].

In this paper, we are more precisely interested in the performance of the Cuckoo search algorithm as a matching strategy, more then it is a question of building a discriminant features of the sought object. The remainder of this paper is organized as follows: after a brief introduction of the reidentification issues and challenges in the first section, we detail the adopted parametric representation in Sect. 25.2. Section 25.3 provides details on the projection of the cuckoo search on the reidentification problem, while Sect. 25.4 provides the obtained experimental results. We conclude by providing an analysis of the obtained results and a glimpse on the future works triggered by this research outcomes.

25.2 Object Representation

In the world of multiple object tracking in a camera network with multiple fields of view, it is common to speak about two set of profiles interacting during the matching process. A query set represents a group of profiles to be identified, and a candidate set composed of a chosen set of profiles of which the identity is known, and that should match the ones in the query set. A simple temporal reasoning can simplify the matching task by roughly estimating the requested transition time between dif-

ferent cameras for a specific query profile [11]. Such a reasoning would improve the quality of the candidate set based on a pure temporal preselection. The restriction would limit the set prior to the computation of any visual feature. State of the art approaches usually use the same set of visual characteristics and comparison metrics in order to map the queries to their corresponding candidates, which is not the most efficient approach. Our currently proposed approach also adopts this behavior as per its intuitive nature, its computation cost as well as the programmatic simplicity. Figure 25.3 presents a challenging scenario for such a reasoning: two profiles (a1) and (b1) are observed in camera 1, with their corresponding candidate set (a2) and (b2), both observed in camera 2. The red box highlights the reidentification algorithm's output as a corresponding profile for the query image. As can be observed in the image, the color feature can be used to identify the query profile correctly in the set (a), while the same feature would not be as much efficient in the set (b). The algorithm would have to rely on another feature such as the shape or texture. The features would have to be correctly balanced, according to the input image.



Fig. 25.3 Matching process between query samples and candidate profiles

Persons reidentification is generally based on the following characteristics:

- **Colour:** generally used for its intuitive and discriminant capability [4]. Objects are identified by their color histograms in the RGB (Red-Green-Blue) color space or HSV (Hue-Saturation-Value) color space.
- **Shape:** one of the most known shape signature is the Histograms Of Gradients (HOG) [15, 21].
- **Texture:** often represented by the Gabor filters [17], differential filters [7], Haar based representations [1] or even co-occurrences matrix [21].
- **Points of interest:** varies between the SURF (Speeded Up Robust Features) points [5] and the SIFT (Scale-Invariant Feature Transform) points [8], depending on the domain of application and the speed/accuracy requirement.
- **Regions of interest:** introduced in [4, 15].

Generally, the region based methods tend to split the human body into multiple connected regions, and then extract different features for each body part. These

methods tend also to represent each individual by a complex combination of different weighted characteristics: we cite [14] which introduces the bio-inspired characteristics idea, as well as [10] which automatically discovers a semantic attributes anthology by training its classifier over publicly available databased. One particularly interesting approach is the one introduced by Li et al. [11]: each profile is defined by a combination of 5 low-level features, forming a descriptive vector by concatenating all resulting features. The used features in this approach are dense color feature, dense SIFT, HOG descriptor, Gabor descriptor and the LBP signature. The size of the resulting vector is then condensed using the Principal Component Analysis (PCA) to retain 90% of the descriptor’s energy, and then normalized.

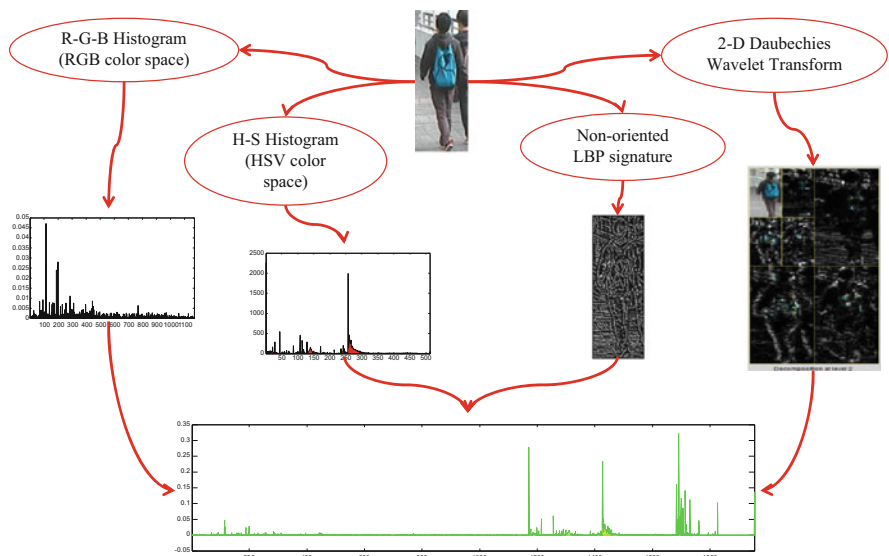


Fig. 25.4 Adopted feature combination for building a lightweight discriminative descriptor for reidentification purpose in a MOT scenario

This approach surely produces a very robust descriptive vector against most and different types of possible (known) distortions. Nevertheless, we are more interested in the computation cost. Therefore, we chose a lighter descriptor, with a reasonable discriminative rate. Figure 25.4 provides a descriptive schema of the adopted features in the present research. For each profiles, two major aspects are captured: the color and the texture. This combination turned out to be sufficient when trying to build a discriminative representation of a profile in the framework of multiple object tracking. The color component is build using the concatenation of the normalized vector of the color density distribution in the RGB color space as well as the H and S components in the HSV color space. The texture is composed of a two-dimensional

Daubechies wavelet transformation, concatenated to the non-oriented LBP signature of the profile. The concatenation of both color and texture components form the descriptive vector of the object. The discriminative power is then strengthened by a temporal reasoning, restricting our search over a subset of candidates.

Another asset of the adopted representation is the computational cost. Such a combination does not request a high computation time, compared to using the Gabor filters for example to represent the texture appearance, and therefore is suitable for a real-time application. This representation is then somehow similar, but less “crowded” and much faster than the one introduced and adopted in [17]. The later uses an additional color feature in the YCbCr color space, and replaces our texture component by 21 texture filters, composed of Gabor and Schmidt filters applied to the luminance channel.

25.3 Projection of the Modified Cuckoo Search on the Multiple Object Tracking Problem

As we previously explained, there are multiple reidentification methods since this research field has been introduced. These approaches are generally relying on a classification instance selected based on a priorly defined parameters or sometimes on a simple author’s personal preference. The optimization work is then generally performed on the descriptor’s combination level, in order to strengthen the discrimination capacity over a large set of pedestrians. The main aim of such a contribution is its accuracy in terms of reidentification produced by the designed algorithm. Many implementations are also made publicly available for comparison purposes.¹

Nevertheless, regardless of the accuracy provided in terms of false and correct matchings—which we will hereby denote as TP and TN—, such algorithms present a major drawback residing in the algorithm’s speed. In fact, the algorithm’s rapidness is generally omitted in such a problem, and all the attention is focused on the ID-Switch reduction. This leads to very accurate algorithms in terms of reidentification and based on low-level and high-level features, unfortunately with very reduced number of application domains as per its computation complexity. In this present work, We tried to address this problem, by making use of the Modified Cuckoo Search (MCS) [22] algorithm in the correspondence phase. We designed a special context in order to validate the proposed method: the reidentification shall be used in the context of a multiple object tracking problem. The used video includes a mean of 16 moving persons per image [2].

Before computer vision, multiple object tracking has been already explored by multiple projects within the Radar and signal processing community, leading to multiple original works in that field such as [19], which now are the basis of the reidentification field. These works were annotated as Multi-Hypothesis Trackers

¹ Example of reidentification open-source project: <https://github.com/Robert0812/saliencereid> introduced by Zhao et al. [26].

(MHT). The main purpose was to build a set of detection in a crowded tracking environment over a specific time frame, to produce a spacious-temporal tracking of targets generating the detections. The MHT theory results in a tree of possibilities exponentially evolving over time. Thus, further researches investigated on how to filter the resulting nodes of the possibility tree using specific optimization algorithms, and avoiding therefore the classical classification algorithms during the matching phase.

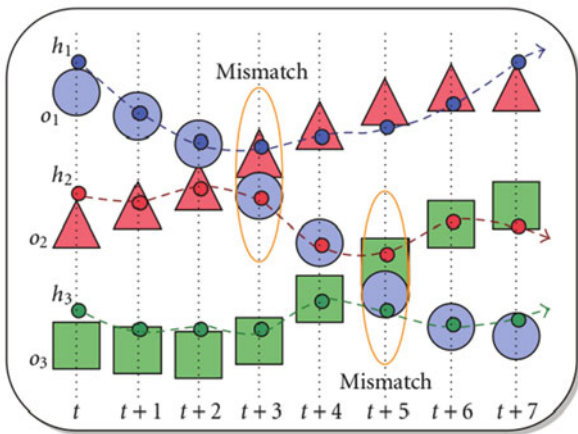


Fig. 25.5 Temporal associations of three objects $o_{i1 < i < 3}$ and three tracking hypotheses $h_{i1 < i < 3}$. Two mismatches are highlighted (source [9])

However, the multiple target tracking implies a temporal association issue between different detections as illustrated in Fig. 25.5. Since we are aiming for a first validation of the theory in a mono-view tracking context, each ID-Switch (i.e. mismatch) would be propagated over time and would induce a high error rate. To avoid such a situation, we included a validation system for ambiguous results based on the normalized cross correlation algorithm [25].

We start our approach the creation of the descriptors database. In fact, in order to take different appearances of the same object into consideration, we use a descriptors database as illustrated in Fig. 25.6. The database is initialized using the first frame of the video. Based on the provided Ground Truth location data, the algorithm considers a bounding box around each moving person. This procedure replaces a perfect prior detection, since each misdetection would result in biased descriptor and therefore induce a poor reidentification performance and thus a high ID-Switch rate. The rectangular bounding box delimits the image region where the feature descriptor will be computed. This descriptor is stored afterwards in a matrix that we call a descriptors database. This database hold the objects feature descriptor along with its ID. Each object is then associated to a representative and randomly initialized color in the RGB color space. After this process has been applied to all objects in the first frame, the algorithm sorts the descriptors database based on the objects IDs.

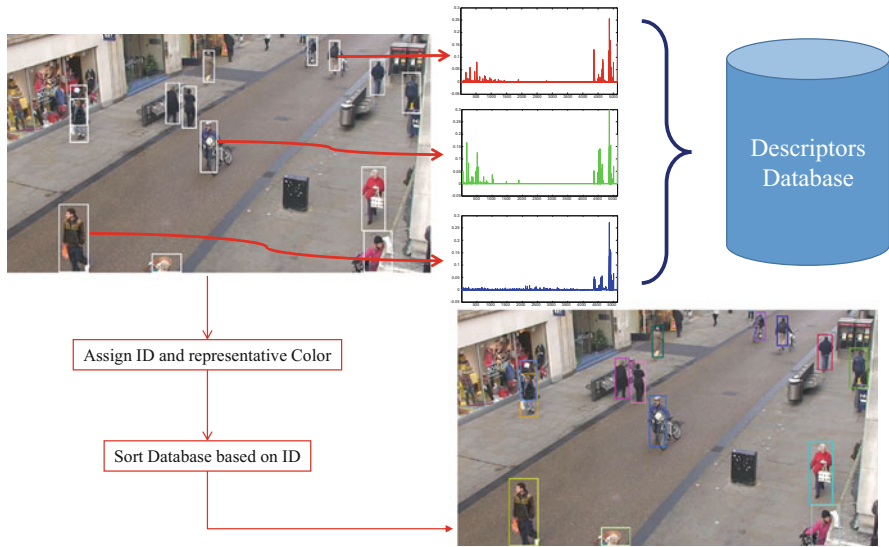


Fig. 25.6 Initialization process of the descriptors database

The Cuckoo Search based reidentification algorithm starts then from the second image. The descriptors database is then recalled at each reidentification process. Figure 25.7 describes the functioning schema of the proposed approach. We will then detail each of the phases described in the figure. In fact, the algorithm is composed of following stages:

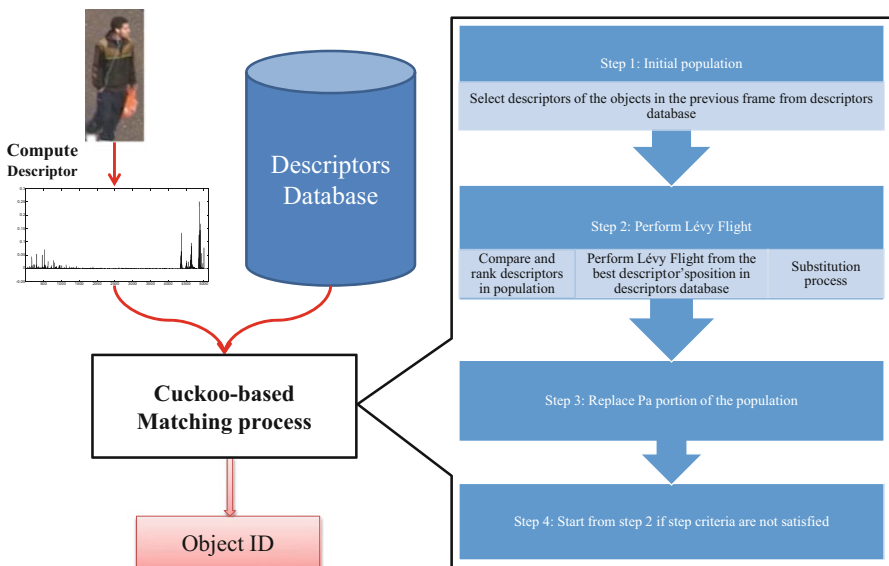


Fig. 25.7 Functional diagram of the MCS based reidentification approach

0. Start of a request: each detection (provided in the ground truth file) is considered as a query profile. The candidate set is then picked up from the descriptors database as an initial population for the cuckoo search algorithm
1. Creation of the initial population: the initial population, as explained in [24], is composed of a set of object descriptors instead of simple histograms. It is initialized using the descriptors of the objects previously detected or identified in the previous frame. This concept illustrates the temporal selection theory as explained in the previous section. The probability of having an object in the current frame that was already present in the previous one is much higher than the probability of seeing an object that already left the scene 100 frames before. Thus, it is more likely to obtain a fast convergence using this roll-back process, then randomly initializing the initial population from the database with no consideration to the time and “maximum velocity constraint” factors.
2. Lévy Flights: once the population is initialized, the matching process takes place. This is done by comparing the query’s descriptor to the gallery’s (i.e. candidate) using the Euclidean distance between both vectors. The candidates are then sorted based on their similarity measure. The smaller the distance, the bigger the similarity between the query and the gallery test. The best candidate in the initial population is picked up, and starting from his position in the descriptors database, a Lévy Flight is performed. It is worth reminding that the database is sorted using IDs. This implies that the Lévy Flights would help explore the neighborhood and therefore select another descriptor of the same individual that might refine the similarity with the query image, while keeping an eye on remotely located individuals, and thus avoiding getting stuck on a local minimum. The selected descriptor using the Lévy Flights is then compared to a randomly chosen descriptor from the initial population. If the later is of a lower quality, it is substituted by the selected one using the Lévy Flights. Since we are using the modified version of the Cuckoo Search as introduced by Walton et al. [22], the position of the Lévy Flights candidate is calculated based on the best obtained element using Eq. (25.1):

$$X_{(t+1)} = X_t + \sqrt{\frac{1}{i}} \oplus \text{Lévy}(\lambda), \quad (25.1)$$

Where $X_{(t)}$ refers to the candidate’s position in frame t , $\text{Lévy}(\lambda)$ is a Lévy flight which provides a random walk based on a random step length drawn from a Lévy distribution

$$\text{Lévy}(\lambda) \approx t^{-\lambda}, \quad (1 < \lambda \leq 3)$$

Using the number of performed generations as an inversed multiplicative scalar $\sqrt{\frac{1}{i}}$ to the generated Lévy step helps transforming the search from an exploration pattern to a refinement.

3. Discovery by the host bird: next to the Lévy Flights, we illustrate the discovery phase, where the host bird discovers the alien egg (i.e. the cuckoo’s egg) and reacts accordingly. This phase is represented in the algorithm by a blind substitution process of a portion of the initial population with a new randomly generated portion picked up from the descriptors database. The portion’s size is of P_a and includes the least similar descriptors to the sought one. In order to locate them,

the population is sorted at the beginning of this phase based on the similarity measure of all population's descriptors to the target profile.

4. Stop Criteria evaluation: by the end of the substitution phase, the algorithm checks whether one or more of the stop criteria is met. If yes, the algorithm stops assuming a total convergence, and hands out the ID of the last best descriptor obtained. Stop criteria are defined as follows:

- A maximum number of iterations are performed without obtaining a satisfying descriptor.
- An considerable number of iterations are performed without any remarkable change similarity measure of the best element.
- A descriptor with a satisfying similarity has been reached.

The last obtained descriptor as well as its Euclidean distance ore provided as output in order to check if the obtained results are classified as ambiguous case or not.

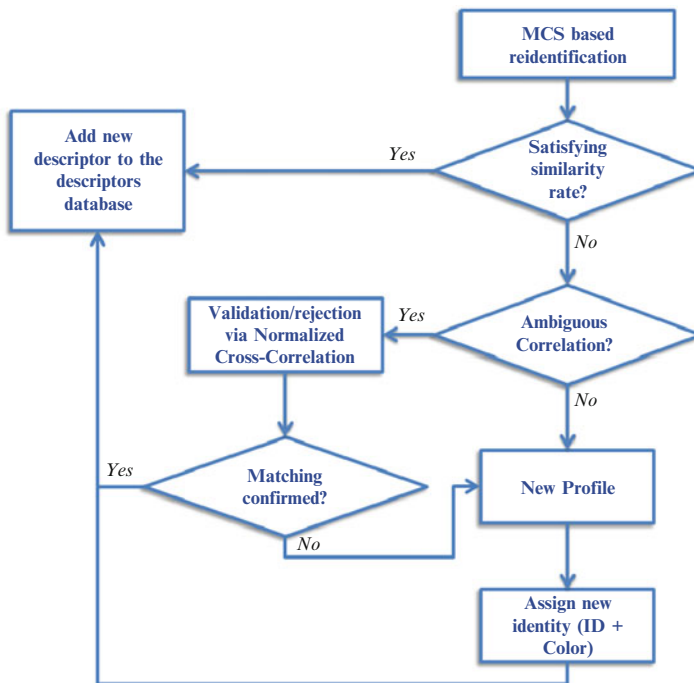


Fig. 25.8 Validation schema of the matching result driven by the MCS

After a total convergence of the algorithm, the obtained similarity measure is checked. The functional diagram describing the different possibilities is exhibited in Fig. 25.8. The algorithm checks if the obtained similarity measure driven by the MCS refer to a perfect match between the query and the best obtained descriptor. Where applicable, no further validation is needed and the query descriptor is inserted in the descriptors database, and the ID of the best matching descriptor is assigned

to the query descriptor in the database. The database is then sorted again based on the IDs in order to gather different descriptors of one same person in the same area of the database. This is an essential step in order to preserve the advantage provided by the Lévy Flights. In ambiguous cases where the similarity measure is not totally convincing, the algorithm proceeds with an ID confirmation, which is carried out by the normalized cross correlation. If the match is confirmed, the algorithm proceeds as in the first case by inserting the new descriptor with its corresponding ID in the database. Otherwise, the match is rejected and the algorithm considers the query descriptor as a new appearance corresponding to a new person in the video, and assigns a new identity and representing color in the database to it. The same conclusion is drawn when the provided similarity measure from the MCS matching process is very low

25.4 Experimental Results

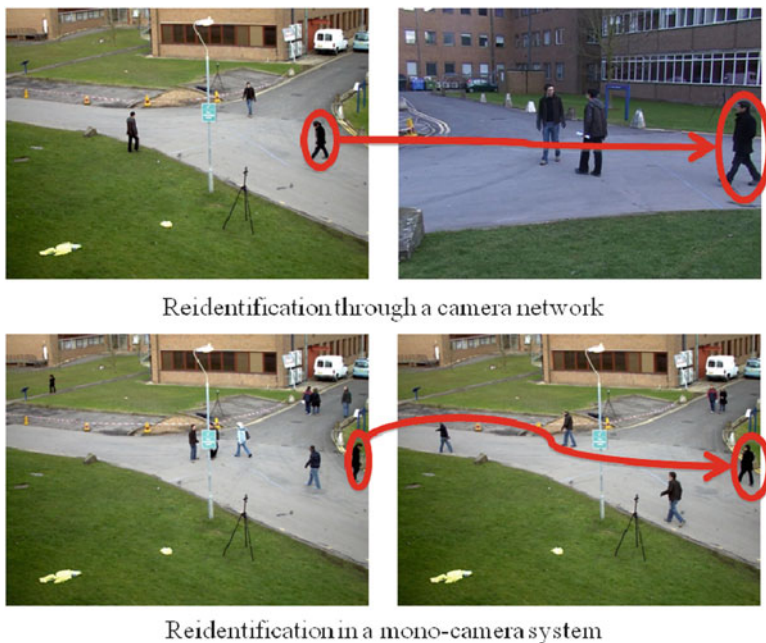


Fig. 25.9 Reidentification scenario in mono-camera VS multi-camera systems

It is worth mentioning that this present work is a first milestone of which the main aim is to validate the reidentification theory using the cuckoo search algorithm. The validation has been carried out in a mono-camera environment and is meant to be extended to a multi-camera network system as described in Fig. 25.9. In order to validate our approach, we adopted the ID-switch metric as introduced in [9]. Since we completely rely on the provided ground truth data, assuming a perfect prior detection, we see no use of computing the MOTA and MOTP as they are more for the accuracy and precision of detection.

In fact, due to the recent extensive growth in the number of approaches introduced for multiple object tracking, and because most of them rely on completely original ideas and very special logic, comparison between different algorithms was becoming a hard task. The lack in common metrics made it hard to compare different results of algorithms, even when run using the same test dataset. One of the most remarkable and successfully and now extensively used contributions was formulated in [9] and labeled The CLEAR MOT Metrics, where authors introduced and developed different metrics covering different aspects of the multiple object tracking area. In this paper, authors explicitly provided two intuitive metric definitions to allow comparison between trackers characteristics, as well as an implicit description of a third metric to assess the algorithms ability to correctly and consistently label tracked objects along their appearance. Following definitions were provided:

- The multiple object tracking precision (MOTP): which evaluates the tracker's ability to precisely locate an object compared to its ground truth position, regardless of whether the object is correctly recognized. It is calculated by summing the error in the estimated position, averaged by the total number of correspondences established.
- The multiple object tracking accuracy (MOTA): gives an idea on the algorithm's capability to correctly detect moving objects in a frame, and consistently keep track of their trajectory. It sums up all misdetections such as mismatches, false positives (regions of the image labeled as moving objects while belonging to the background) and misses in the sequence. It is somehow similar to the widely used Word Error Rate (WER) used in speech recognition.
- ID-switch: is a metric that tackles tricky situations where the same reidentification error can be made, while counted differently if counting incorrect matches along the sequence. Assuming O_1 is incorrectly identified in two frames by tracker 1, while incorrectly labeled in four frames by tracker 2, the number of mismatches for tracker 1 equals 2 and 4 for tracker2, while they both switched O_1 's ID once. Therefore, the ID-switch value for both algorithm's equals 1.

In all tests run in this section, the stop criteria were configured as follows:

1. A maximum number of 3 iterations are performed without obtaining a satisfying descriptor.
2. 20 iterations are performed without any remarkable change similarity measure of the best element. The end of the section provides an evaluation of the configuration impact on the algorithm's performances when this parameter is changed.
3. A descriptor with a similarity measure of 0.4 has been reached. This value is experimentally obtained after trying multiple different values.

The first set of experiment is designed in order to validate the algorithm's capacity to recognize a patch after a long occlusion period. To do so, we used a video from the CAVIAR² database, with 1387 frame of which size is $288 * 384$. Figure 25.10 shows the obtained visual results of this experiment.

² Dataset Available at : <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/>.

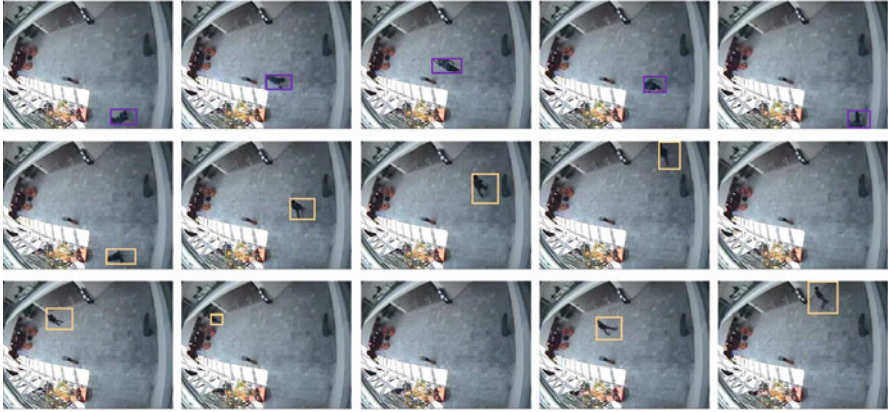


Fig. 25.10 Reidentification results of multiple persons in presence of long time occlusions

Once the first moving person dressed in a blue shirt appears, the algorithm computes a descriptor patch and tracks the person correctly and successfully. As he leaves the stage, another person with a black shirt appears. The algorithm computes the new descriptor and compares it to the already existing patch in the database. The algorithm correctly distinguishes between both profiles, and assigns a new ID to the person with the black shirt. It is worth mentioning that the second person enters the scene from the same spot where the first person disappeared. The second moving person crosses the scene, then disappears for another 60 frames. Once back, the algorithm correctly recognizes this person and tracks is successfully. Another occlusion is observed for more than 100 frames, and the person is correctly recognized again as can be seen in the third line of Fig. 25.10.

In order to test the algorithm's performances in a crowded scene, we used a video from the Town Center database which is provided by the Oxford university. The used video is made of 7500 frames, with an HD resolution of $1080 * 1440$. An average of 16 moving person per frame are present in this video [2]. With such a density, one ID-switch represents a missing rate of 0.0625% ($\frac{1}{10}$), which makes it $6.25 * 10^{-4}\%$ in a 100 images long sequence. Figure 25.11 shows a visual result of the algorithm's performances. The proposed method is able to build the trajectory of multiple moving persons with a very reduced missing rate, at very high speed. In fact, the success rate is around 99% in 128 s/100 frames, which is around 1.28 per image, and thus 0.08 s per profile, which clearly overtakes the state of the art approaches. When allowing more iterations per frame (around 150 iteration per frame), the algorithm is able to lower the mismatches to 4 ID-switch in a 1600 matching operation.

And since sample images from a tracking performances are not convincing enough, we carried out a specific experiment to highlight the impact of the maximum number of iteration per frame over the algorithm's convergence. Figure 25.12 provides a visual aspect of the obtained results. We conclude that the higher the number of allowed iteration per frame, the more accurate the reidentification results, which makes perfect sense since we allow a total convergence to the algorithm.



Fig. 25.11 Reidentification performances in a crowded scene with an average of 16 profile per image

Also, the increase shown in the execution time is logically inversely proportional to the mismatch rate. Note that up to a certain threshold (in this case 150), the ID-switch value becomes constant as the algorithm reaches its accuracy limit. Therefore allowing more iterations per frame would only affect the execution time due to superficial computation.

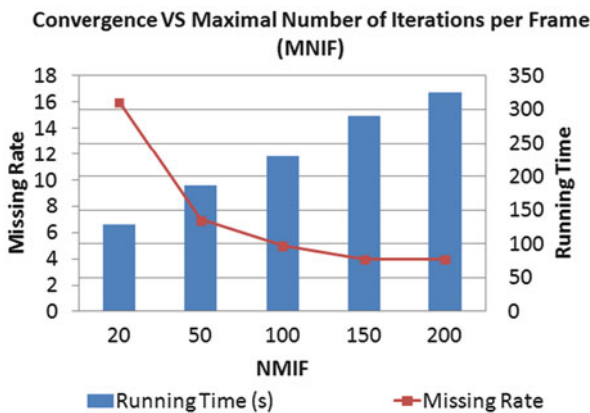


Fig. 25.12 Relationship between the ID-switch rate, the execution time and the maximum number of allowed iteration per frame

Nevertheless, the designed architecture presents a major drawback. Since around 16 descriptors are stored in the descriptors database at each image, an important storage memory is necessary for each execution. And because of its definition, each descriptor is a concatenation of multiple vectors, and therefore requires a considerable memory size, which presents the algorithm’s handicap. The first set of images are rapidly processed, and because a sorting operation is performed at each iteration, it becomes time consuming as the database grows bigger. The same concept is valid for the color association process at the visualization step. Figure 25.13 shows the evolution of the execution time per image. The spikes correspond to the execution of the normalized cross-correlation validation, which is a time consuming operation. The increase of the requested processing time highlights the major drawback of storing all used descriptors, preventing therefore a correct processing of the entire 7500 HD images. The algorithm provides acceptable results up to 300–350 frames.

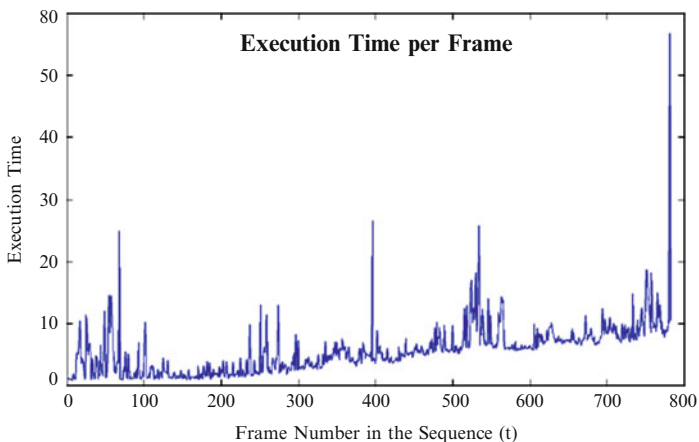


Fig. 25.13 Increase of the execution time during the processing of an HD video due the growth of the descriptors database

25.5 Conclusion

In this paper, we introduce a novel approach for person reidentification using the modified version of the cuckoo search algorithm. After a brief introduction to the problematic, we presented the state of the art of the used parametric representation in that field. The projection of the cuckoo search algorithm into the reidentification problem was then detailed, including the justification of the used parametric representation. As a validation step to our proposed architecture, we used a lightweight combination of color and texture representation of the query object. The validation step is carried out in a mono-camera system as a matching process in a tracking performance. A descriptors database is considered in order to take into consideration multiple possible appearances of the same moving person for accuracy purpose. The algorithm starts by initializing the database using the Ground Truth file, and at each reidentification attempt, the algorithm performs a temporal selection by picking up the descriptors of the previous frame from the database. A Lévy flight is then performed around the best descriptor in the database, and the candidate compared to a randomly selected element, which is substituted in case the similarity is increased. A portion of the initial population is then replaced with a probability P_a . After convergence, the obtained ID is confirmed or rejected using the normalized cross-correlation method. The obtained results are promising while a major drawback was highlighted: storing all descriptors in the database slows the algorithm's performances as the database grows. Future work will investigate on how to efficiently sort the database and eliminate useless descriptors after a certain amount of time. The cleanup process shall enhance the algorithm's performance and make it sustainable along very long video sequences.

References

1. S. Bak, E. Corvee, F. Brémond, M. Thonnat, Person re-identification using haar-based and DCD-based signature, in *2010 Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (IEEE, New York, 2010), pp. 1–8
2. B. Benfold, I. Reid, Stable multi-target tracking in real-time surveillance video, in *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New York, 2011), pp. 3457–3464
3. E.D. Cheng, M. Piccardi, Matching of objects moving across disjoint cameras, in *2006 IEEE International Conference on Image Processing* (IEEE, New York, 2006), pp. 1769–1772
4. M. Farenzena, L. Bazzani, A. Perina, V. Murino, M. Cristani, Person re-identification by symmetry-driven accumulation of local features, in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New York, 2010), pp. 2360–2367
5. N. Gheissari, T.B. Sebastian, R. Hartley, Person reidentification using spatiotemporal appearance, in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2 (IEEE, New York, 2006), pp. 1528–1535
6. S. Gong, M. Cristani, S. Yan, C.C. Loy, *Person Re-identification* (Springer, London, 2014). doi:10.1007/978-1-4471-6296-4
7. D. Gray, H. Tao, Viewpoint invariant pedestrian recognition with an ensemble of localized features, in *Computer Vision—ECCV 2008* (Springer, Berlin, 2008), pp. 262–275
8. K. Jungling, C. Bodensteiner, M. Arens, Person re-identification in multi-camera networks, in *2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (IEEE, New York, 2011), pp. 55–61
9. B. Keni, S. Rainer, Evaluating multiple object tracking performance: the CLEAR MOT metrics. *EURASIP J. Image Video Process.* **2008**, 1–10 (2008)
10. R. Layne, T.M. Hospedales, S. Gong, Re-id: hunting attributes in the wild, in *Proceedings of the British Machine Vision Conference (BMVC)* (2014)
11. W. Li, R. Zhao, X. Wang, Human reidentification with transferred metric learning, in *Computer Vision—ACCV 2012* (Springer, Berlin, 2013), pp. 31–44
12. W. Li, R. Zhao, T. Xiao, X. Wang, Deepreid: deep filter pairing neural network for person re-identification, in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014), pp. 152–159. doi:10.1109/CVPR.2014.27
13. C.C. Loy, T. Xiang, S. Gong, Multi-camera activity correlation analysis, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009)* (2009), pp. 1988–1995. doi:10.1109/CVPR.2009.5206827
14. B. Ma, Y. Su, F. Jurie, Bicov: a novel image representation for person re-identification and face verification, in *Proceedings of the British Machine Vision Conference* (BMVA Press, Guildford, 2012), pp. 57.1–57.11. <http://dx.doi.org/10.5244/C.26.57>
15. O. Oreifej, R. Mehran, M. Shah, Human identity recognition in aerial images, in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New York, 2010), pp. 709–716
16. W. Ouyang, X. Chu, X. Wang, Multi-source deep learning for human pose estimation, in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New York, 2014), pp. 2337–2344
17. B. Prosser, W.S. Zheng, S. Gong, T. Xiang, Q. Mary, Person re-identification by support vector ranking, in *Proceedings of the British Machine Vision Conference (BMVC)* vol. 3 (2010), p. 5
18. Y. Raja, S. Gong, Scalable multi-camera tracking in a metropolis, in *Person Re-identification, Advances in Computer Vision and Pattern Recognition*, ed. by S. Gong, M. Cristani, S. Yan, C.C. Loy. (Springer, London, 2014), pp. 413–438. doi:10.1007/978-1-4471-6296-4-20
19. D.B. Reid, An algorithm for tracking multiple targets. *IEEE Trans. Autom. Control* **24**(6), 843–854 (1979)

20. J. Sanchez-Riera, Y.S. Hsiao, T. Lim, K.L. Hua, W.H. Cheng, A robust tracking algorithm for 3d hand gesture with rapid hand motion through deep learning, in *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)* (IEEE, New York, 2014), pp. 1–6
21. W.R. Schwartz, L.S. Davis, Learning discriminative appearance-based models using partial least squares, in *2009 XXII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)* (IEEE, New York, 2009), pp. 322–329
22. S. Walton, O. Hassan, K. Morgan, M. Brown, Modified cuckoo search: a new gradient free optimisation algorithm. *Chaos Solitons Fractals* **44**(9), 710–718 (2011). <http://dx.doi.org/10.1016/j.chaos.2011.06.004>. <http://www.sciencedirect.com/science/article/pii/S096007791100107X>
23. X. Wang, G. Doretto, T. Sebastian, J. Rittscher, P. Tu, Shape and appearance context modeling, in *IEEE 11th International Conference on Computer Vision, ICCV 2007* (IEEE, New York, 2007), pp. 1–8
24. X.S. Yang, S. Deb, Cuckoo search via lévy flights, in *Proceedings of World Congress on Nature & Biologically Inspired Computing, NaBIC 2009* (IEEE, New York, 2009), pp. 210–214
25. X. Yin, Y. Sun, S. Song, X. Ma, A target tracking algorithm based on optical transfer function and normalized cross correlation, in *The Proceedings of the Second International Conference on Communications, Signal Processing, and Systems* (Springer, Berlin, 2014), pp. 1021–1027
26. R. Zhao, W. Ouyang, X. Wang, Unsupervised saliency learning for person re-identification, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013)

Chapter 26

Facing the Feature Selection Problem with a Binary PSO-GSA Approach

Malek Sarhani, Abdellatif El Afia, and Rdouan Faizi

Abstract Feature selection has become the focus of much research in many areas where we can face the problem of big data or complex relationship among features. Metaheuristics have gained much attention in solving many practical problems, including feature selection. Our contribution in this paper is to propose a binary hybrid metaheuristic to minimize a fitness function representing a trade-off between the classification error of selecting the feature subset and the corresponding number of features. This algorithm combines particle swarm optimization (PSO) and gravitational search algorithm (GSA). Also, a mutation operator is integrated to enhance population diversity. Experimental results on ten benchmark dataset show that our proposed hybrid method for feature selection can achieve high performance when comparing with other metaheuristic algorithms and well-known feature selection approaches.

Keywords Feature selection • Particle swarm optimization • Gravitational search algorithm • Machine learning • Metaheuristics

26.1 Introduction

Nowadays, machine learning approaches have gained much attention and wide applications in solving non-linear and complex problems. In particular, choosing the most suitable subset of descriptors among a large number of datasets is one of the most interesting and challenging steps of solving these kind of problems. Traditional statistical methods such as principal component analysis (PCA) and factor analysis may fail to deal with these problems because it supposes in their native form a linear relationship among variables. Furthermore, these problems may evolve a huge number of datasets (big data). Therefore, approaches such as feature selection (FS) have been proposed to reduce dimensionality in complex problems. Feature selection

M. Sarhani (✉) • A. El Afia

Department of Informatics and Decision Support, Mohammed V University, Rabat, Morocco

e-mail: malek.sarhani@um5s.net.ma; rdfaizi@gmail.com

R. Faizi

ENSIAS - Mohammed V University, Rabat, Morocco

e-mail: a.elafia@um5s.net.ma;

© Springer International Publishing AG 2018

L. Amodio et al. (eds.), *Recent Developments in Metaheuristics*,

Operations Research/Computer Science Interfaces Series 62,

DOI 10.1007/978-3-319-58253-5_26

or feature subset selection aims at identifying the most relevant input (predictors) within a dataset. FS may improve the performance of the predictors by eliminating irrelevant inputs, it may also achieves data reduction for accelerated training and increases computational efficiency [21]. Recent studies have mentioned other advantages of the FS task.

There are three main approaches for FS [8]. Filter ones are based on information theory. Wrapper and embedded approaches utilize a machine learning algorithm to score subsets of features according to their predictive capability. In embedded approaches, the selection is done in the training process while in wrappers the machine learning classifier is used as a black box. Therefore, we can improve the searching part in wrappers separately from the classifier. Furthermore, wrappers can promise better results than filter approaches. But, they are more computationally demanding [20]. Therefore, we have chosen in this paper to use wrapper methods which are the most commonly used approach within metaheuristics as we can evaluate the searching algorithm separately. Wrappers have been popularized by Kohavi and John [14]. Wrapper approaches hybridize between a machine learning method for classification and an algorithm adapted to search in the space of feature subsets which is a binary space [8].

Two widely used wrapper feature selection algorithms are forward selection and backward selection. The first one consists in starting without any feature. In the second one, the wrapper method starts with all candidate variables. Also, these search algorithms can mainly categorized into one of the following categories: exact methods, greedy sequential subset selection methods, and metaheuristics. The use of metaheuristics in this context is justified by the fact that FS is an NP-hard problem. The fitness function that has to be optimized within the wrapper approach corresponds generally to the error rate of the used classifier [12]. For certain problems, the number of features can also be considered as a second objective to be minimized [29].

Over the last decades, various population based metaheuristic algorithms have been introduced and applied successfully to solve a wide range of the optimization problems in both continuous and binary search spaces. Particle swarm optimization (PSO) and gravitational search algorithm (GSA) are two popular metaheuristics which are initially designed for continuous optimization problems and are extended to binary search problems. That is, the binary PSO (BPSO) was proposed by Kennedy and Eberhart [13] and the binary GSA (BGSA) was recently defined by Rashedi et al. [23].

PSO and GSA are effective optimization algorithms. But, they suffer both from two main problems: premature convergence (these algorithms can be trapped easily into a local optima) and slow convergence when the best solution found is near to the optimum solution. These problems, are more frequents and have a more impact when treating binary problems. Thus, one of the main challenges within these algorithms is to improve its capability for both global exploration and local exploitation abilities. Hybridization of metaheuristics is an active research trend that may help to achieve these goals and especially the first one. Furthermore, operator such as mutation and local search can be added to have a stronger exploration ability and then

enhance the second feature. Therefore, in this paper, we examine the effectiveness of a binary hybrid metaheuristic combining PSO and GSA and enhanced by a mutation operator for the wrapper approach of FS.

The rest of the paper is organized as follows: in the next section, we outline the related works. In Sect. 26.3, a brief overview of BPSO and BGSA is provided. Moreover, our hybrid algorithm is described in addition to its contribution to FS. Section 26.4 presents the obtained results for the experiments. Finally, we conclude and present perspectives to our work.

26.2 Literature Review

Feature selection has been applied in different fields. It is most used in medical problems which may evolve, sometimes thousands of variables [8]. The fitness function can be determined in this case by looking for a trade-off between specificity and sensitivity [7]. Another common application of FS is in forecasting time series. Indeed, FS can enable forecasting machine learning approaches to deal with real world dataset which may contain missing values, outliers, redundant variables and may evolve a large number of historical values [10]. In general, FS can be used in real problems from different fields which may evolve a non-linear relationship among variables or large dataset.

In regards of the methods used in literature for FS, we have to mention first that a number of papers adapted or extended the statistical methods like principal component analysis (PCA) as it can be adequate for FS, see for instance [17]. Furthermore, the mutual information approach is a powerful mechanism which have been widely investigated for this application [27].

Concerning using metaheuristic in wrapper approaches. The BPSO has been applied for FS in different forms, the most used form is the one proposed by Kennedy and Eberhart [13] as presented in the following section. It has been applied for instance in [26] and [29]. This transformation has been used also for BGSA in [4]. Another way has been presented for example by Talbi et al. [25] which adopted the Geometric PSO (GPSO) to the FS problem and compared it to the genetic algorithm (GA). BGSA has been also used for FS in its native form presented by Rashedi et al. [23], see for instance [2]. Other metaheuristics such as simulated annealing and ants colony optimization can be used for FS: a review on the integration of metaheuristics in FS can be found in [5].

Recently, Hybridization of PSO and GSA has been become promising to solve continuous problems as in [6], [11] and [1]. But, these combination doesn't always increase the performance of the algorithm as in [18] in which the GSA technique has given a better results than PSOGSA.

Regarding avoiding premature convergence for BPSO and BGSA, [1] proposed to hybridize the BPSO with local search. Furthermore, [30] used a mutation operator for this purpose, [9] chosen linear chaotic map to enhance the diversity of the search

space to BGSA and [16] has hybridized mutation algorithm with local search to deal with the FS problem.

Also, we have noticed that a binary form of PSOGSA has just been published in [19]. The combination proposed in the mentioned paper consists in integrating the global best effect of PSO in the classical BGSA. The authors assumed that this combination may lead to better convergence rate in optimizing their used benchmark functions. A critical view of this approach is provided in Sect. 26.3.4.

At the end of this section, we can conclude on one hand that the combination of PSO and GSA may improve in most cases the convergence rate and performance of these algorithms. On the other hand, in order to enhance the diversification of population, it is important to integrate adaptives operators such as mutation and local search.

26.3 The Proposed Binary PSOGSA Approach

In this section, we present the concepts of BPSO and BGSA. Furthermore, we present and explain our approach including the proposed fitness function and its contribution to related works.

26.3.1 The Canonical BPSO Algorithm

The BPSO algorithm was introduced by Kennedy and Eberhart [13] to enable PSO to operate in discrete and binary search spaces. It follows the same approach of the canonical PSO. Each particle i has two vectors: the velocity vector and the position vector. The particles are updated according to its previous best position and the entire swarm previous best position. That is, particle i adjusts its velocity v_i and position x_i in each generation t according to the following formula:

$$v_i^d(t+1) = wv_i^d(t) + c_1r_1(p_i^j(t) - x_i^j(t)) + c_2.r_2.(p_g^j(t) - x_i^j(t)) \quad (26.1)$$

where $v_i^j(t)$ and $x_i^j(t)$ correspond to the d th dimension of velocity and position vectors of the particle i . $p_i^j(t)$ represents the best previous position of particle i . $p_g^j(t)$ represents the best position among all particles in the population. r_1 and r_2 are two independently uniformly distributed random variables. c_1 and c_2 are the acceleration factors and w is the inertia weight.

In BPSO, the velocities are considered as a probability that the particle will change to one. This transformation is done using the sigmoid function (sig) which is defined as follows:

$$v_i^d(t) = \text{sig}(v_i^j(t)) = \frac{1}{1 + e^{-v_i^j(t)}} \quad (26.2)$$

$$x_i^j(t+1) = \begin{cases} 1 & \text{if } \text{rand}_i \leq \text{sig}(v_i^j(t+1)) \\ 0 & \text{else} \end{cases} \quad (26.3)$$

where rand_i is a uniform random variable in the interval $[0,1]$.

26.3.2 The Classical BGSA Algorithm

Gravitational search algorithm (GSA) is a recent metaheuristic algorithm based on the law of gravity and mass interactions. The binary version of GSA (BGSA) has been first published by Rashedi et al. [23].

The first step is to initialize position and velocities for all agents (particles). Secondly, we adapt GSA parameters according to Eqs. (26.4), (26.5), and (26.7)–(26.11) (the description of GSA parameters can be found for instance in [23]):

$$\text{best}(t) = \min_i \text{fit}_i(t) \quad (26.4)$$

$$\text{worst}(t) = \max_i \text{fit}_i(t) \quad (26.5)$$

$$m_i(t) = \frac{\text{fit}_i(t) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)} \quad (26.6)$$

$$M_i(t) = M_{ii}(t) = \frac{m_i(t)}{\sum_{j=1}^N m_j(t)} \quad (26.7)$$

$$F_{ij}^j(t) = G \cdot \left(\frac{M_{pi}(t)M_{aj}(t)}{R_{ij}(t) + \varepsilon} \right) \cdot (x_j^j(t) - x_i^j(t)) \quad (26.8)$$

$$F_i^j(t) = \sum_{j \neq i} \text{rand}_i F_{ij}^j(t) \quad (26.9)$$

$$a_i^j(t) = \frac{F_i^j(t)}{M_{ii}(t)} \quad (26.10)$$

$$v_i^j(t+1) = \text{rand}_i v_i^j(t) + a_i^j(t) \quad (26.11)$$

where $\text{fit}_i(t)$ is the current fitness of agent i at iteration t . ε is a small constant, $R_{ij}(t)$ is the Euclidean distance between two objects i and j . rand_i is a random variable as defined previously, G is the gravitational constant. p and a are two indices that distinguish between active and passive masses [23].

In a same manner of BPSO, the velocity is considered in BGSA as a probability. But, in GSA, a position updating means a switching between the two possible

values. In other words, it shows the probability of changing the value of $x_i^j(t)$ from “0” to “1” and vice versa. Also, the transformation is done using the tanh function instead of sigmoid function as defined in Eq. (26.12):

$$x_i^j(t+1) = \begin{cases} x_i^j(t) & \text{if } rand_i \leq |\tanh(v_i^j(t+1))| \\ \overline{x_i^j(t)} & \text{else} \end{cases} \quad (26.12)$$

26.3.3 The Aggregative Multi-Objective Fitness Function of FS

To be able to use metaheuristics for FS. The features have to be coded as binary vectors in which the dimension of the vector is the number of features.

Regarding the classification algorithm of the wrapper approach, we choose K-nearest neighbour (KNN) which is commonly used for this purpose. Or, our hybrid metaheuristic algorithm can be combined also with other supervised machine learning algorithms to build the wrapper method.

As described by Xue et al. [29], in addition of the classification error, the performance of a wrapper FS technique can be measured also by the number of features. The trade-off between maximising the classification accuracy and minimizing the number of features can be done in different manners. In this paper, as in [28], we adopt an aggregative fitness function as described in Eq. (26.13):

$$F = \alpha F_1 + \frac{1 - \alpha}{F_2} \quad (26.13)$$

F_1 is the error rate. F_2 is related to the number of features as described in Eq. (26.14):

$$F_2 = n - p \quad (26.14)$$

where n is the complete number of features and p is the number of selected features. The value of α is set to 0.8 because the error rate has to be considered more than the number of features as considered by Vignolo et al. [28] which proposed to adjust α between 0.7 and 0.9.

26.3.4 The Proposed Hybrid Algorithm for FS

For building our approach, we combine firstly PSO and GSA in the same manner used in [19]. Therefore, the update of velocities is done following Eq. (26.15):

$$v(t+1) = \omega v(t) + c_1 r_1 a(t) + c_2 r_2 \cdot (p_g(t) - x(t)) \quad (26.15)$$

Secondly, we have to define the function which can transform a continuous space value into a binary one. [15] have mentioned a number of discretization methods and concluded that the sigmoid function is the most used in the binary context. However, according to [1], these binary extension of PSO suffer from lack of convergence: this problem can be observed for instance if the velocity is near to zero.

Thus, we adopt the tanh transformation (which is used by the classical BGSA) for building our hybrid algorithm. Furthermore, we compare it with the sigmoid function which is used by BPSO (the sigmoid function has been used also for the transformation of BGSA as proposed for instance in [4]).

Also, in order to enhance population diversity, a mutation operator has been added to our hybrid algorithm, we can notice that many mutation techniques have been proposed for this purpose. Here, the uniform mutation is used as we can introduce our algorithm BMP SOGSA. The BMP SOGSA pseudo-code for FS including the uniform mutation is described below (Algorithms 1 and 2) where “ $x_{ij} = \overline{x}_{ij}$ ” symbol in Algorithm 2 refers to the complementary of the binary variable x :

Algorithm 1 Uniform mutation algorithm

Data: Mutation probability (p), the number of particles (n), the number of features (d) and matrix of random numbers $\in [0, 1]$ \mathbb{R} (of $[n \times d]$ dimension)

for $i = 1$ to n **do**

for $j=1$ to d **do**

if $R[i, j] \leq p$ **then**

$x_{ij} = \overline{x}_{ij}$

end if

end for

end for

Results: Updated positions of particles

To better explain the process of using BMP SOGSA in the wrapper FS approach, the flowchart of our proposed approach is depicted in Fig. 26.1: after initializing the BMP SOGSA parameters, the selected features of the complete feature set are updated at each iteration according to the obtained value of BMP SOGSA that minimize the fitness function. This fitness is a trade-off between the KNN classification accuracy and the number of features. This process is repeated until reaching the final solution which contains the selected features.

Furthermore, we investigate the use of a local search (LS) heuristic at the end of the algorithm to refine the solution instead of mutation (as presented in Table 26.3). Local search methods have been used for improving both PSO and GSA performance. Also, [18] used it to improve the continuous PSO GSA.

Our contribution to the paper of Mirjalili et al. [19] which had recently defined the BPSOGSA algorithm is first to adapt it to the FS problem. That is, the code used in their paper (which is available online) isn't yet adapted to the FS problem. Also, in order to improve the diversity of BPSOGSA, we uses the mutation operator. Furthermore, we examine the effect of using local search to refine the obtained solution by BPSOGSA and we evaluate the tanh function for the mapping from continuous

Algorithm 2 BMP SOGSA algorithm

Data: The number of particles (n), the number of features (d) and the initial positions
Initialization: accelerations, positions and velocities of particles
 Set k and t values to 0
while (number of iterations $t \leq t_{max}$ not met) **do**
 Evaluate the fitness $F_i(t)$ of all the individuals
 Determine the best and the worst particles: $best(t)$ and $worst(t)$ according to Eqs. (26.4) and (26.5)
 if ($k \leq 4$) & ($best(t - 1) \leq best(t)$) **then**
 $k \rightarrow k+1$
 else
 Mutate population by uniform mutation algorithm (Algorithm 1)
 Set k value to 0
 end if
 for $i = 1$ to number of particles **do**
 for $d = 1$ to dimension of the problem **do**
 Compute the acceleration $a_i^j(t)$ corresponding to Eq. (26.10) (BGSA algorithm)
 Compute the velocity $v_i^j(t)$ corresponding to Eq. (26.15) (BPSO algorithm)
 Compute the position $x_i^j(t)$ according to Eq. (26.12) using the tanh transformation (BGSA algorithm)
 end for
 end for
 $t \rightarrow t + 1$
end while
Results: The chosen subset based on the best particle in the population and corresponding fitness value

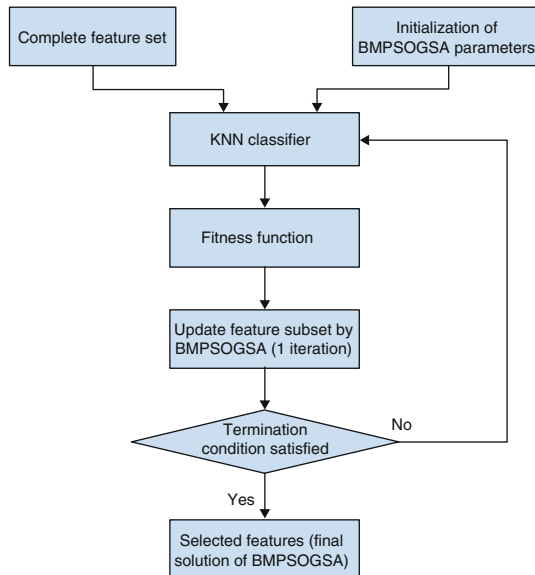


Fig. 26.1 The flowchart of our proposed scheme for FS

to binary spaces by comparing it to the sigmoid function as described in the next section.

26.4 Experiment

26.4.1 Experiment Setup

The selection of the suitable parameters is an important issue in both PSO and GSA. Concerning the parameters c_1 and c_2 , we consider $c_1 = 0.5$ and $c_2 = 1.25$. Regarding the inertia weight w , as commonly used in both continuous and discrete versions of PSO, we choose the time-varying method proposed by Shi and Eberhart [24] which consists on decreasing this parameter according to the following equation:

$$\omega = \omega_{max} - \frac{k}{k_{max}}(\omega_{max} - \omega_{min}) \quad (26.16)$$

k and k_{max} are respectively the current iteration and the maximum number of iterations, the values of w_{max} and w_{min} are respectively set to 0.9 and 0.4. The maximum number of iterations is set to 100, the number of particles (chromosomes for GA and agents for GSA) is set to 40, the mutation rate is set to 0.05 and the random number has been generated by the matlab software.

All algorithms have been executed on a Matlab 8.1 (2013a). Ten datasets widely adopted in benchmarking machine learning methods are used in our experiment. These datasets have taken from the paper of Brown [3], they have different numbers of features, classes and instances as representatives of real problems from different fields as presented in Table 26.1.

Table 26.1 The data sets used in the experiment

Data sets name	Number of features	Number of instances	Classes
breast	30	569	2
congress	16	435	2
heart	13	270	2
ionosphere	34	351	2
krvsnp	36	3196	2
landsat	36	6435	6
lungcancer	56	32	3
parkinsons	22	195	2
semeion	256	1593	10
sonar	60	208	2

26.4.2 Examination of BMPSOGSA for FS

In order to evaluate the efficiency of our approach for FS, we compare its best solution found with those of three other candidate metaheuristics which are: genetic algorithm (GA), BPSO and BGSA. Furthermore, we examine the BPSOGSA when adopting the sigmoid function as a transfer function as in Eq. (26.12) (at the best of our knowledge, BPSOGSA is the unique binary hybridization of PSO and GSA presented in literature).

Therefore, in this section, we depict the obtained values of the fitness function described in Eq. (26.13) in Table 26.2. More details which contains the best solution found (in ‘semeion’, the solution is shown partially) and its corresponding number of features are presented in Table 26.4. Also, to illustrate the mutation and local search (LS) effect on BPSOGSA, Table 26.3 gives a comparative study of BMPSOGSA, BPSOGSA and BPSOGSA-LS. Furthermore, we display in Figs. 26.2, 26.3, 26.4, 26.5 and 26.6 the obtained values by BMPSOGSA in the benchmark dataset at each iteration (Table 26.4).

From Table 26.2, it is clearly shown that the BMPSOGSA gives significantly better results than BPSO, BGSA and BPSOGSA with sigmoid transformation in

Table 26.2 Comparison of BMPSOGSA performance. Bold values are the best obtained results

Data sets	BMPSOGSA	BPSOGSA-Sig	BPSO	BGSA	GA
breast	0.022178	0.028345	0.026577	0.03426	0.022178
congress	0.0347335	0.037808	0.0347335	0.036897	0.0347335
heart	0.10148	0.1131	0.10519	0.11613	0.10148
ionosphere	0.0262336	0.04708	0.042435	0.063746	0.026567
krvskp	0.0191211	0.04	0.024146	0.030022	0.020025
landsat	0.0744114	0.79231	0.0778024	0.078129	0.0721737
lungcancer	0.0047619	0.14	0.05625	0.10645	0.0302632
parkinsons	0.0194872	0.05	0.022284	0.026387	0.0166
semeion	0.0258149	0.0344285	0.029616	0.036646	0.0191769
sonar	0.016667	0.026127	0.026374	0.02099	0.005882

Table 26.3 Examination of mutation and LS effects on BPSOGSA. Bold values are the best obtained results

	BMPSOGSA	BPSOGSA-LS	BPSOGSA
breast	0.022178	0.023765	0.023765
congress	0.0347335	0.0347335	0.0347335
heart	0.10148	0.10148	0.10148
ionosphere	0.0262336	0.026567	0.026567
krvskp	0.0191211	0.017272	0.0192584
landsat	0.0744114	0.073293	0.073293
lungcancer	0.0047619	0.030128	0.302632
parkinsons	0.0194872	0.033846	0.0378089
semeion	0.0258149	0.020937	0.020937
sonar	0.016667	0.005	0.007242

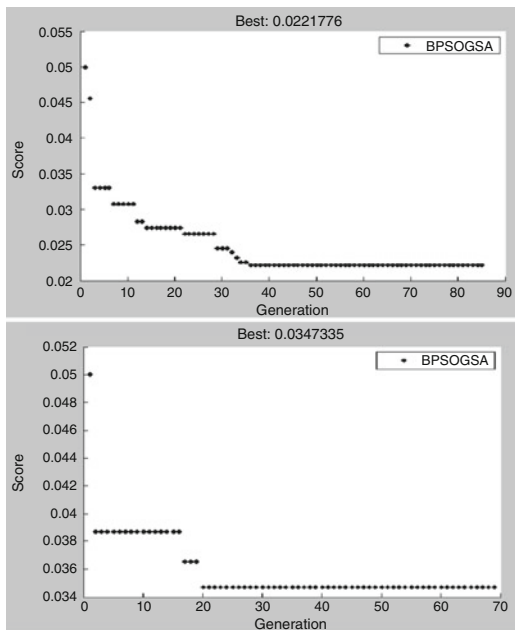


Fig. 26.2 Results obtained for breast and congress dataset for BMPSOGSA

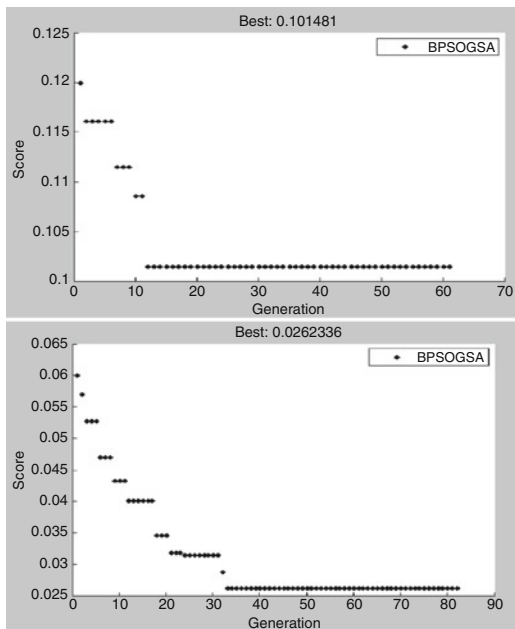


Fig. 26.3 Results obtained for heart and ionosphere dataset for BMPSOGSA

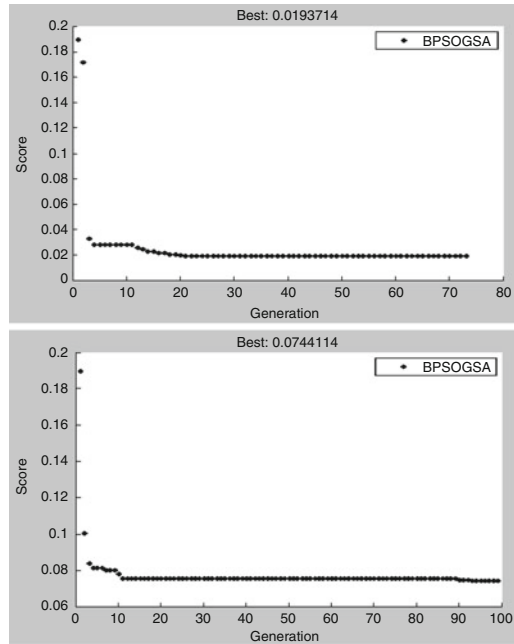


Fig. 26.4 Results obtained for krvsdp and landsat dataset for BPSOGSA

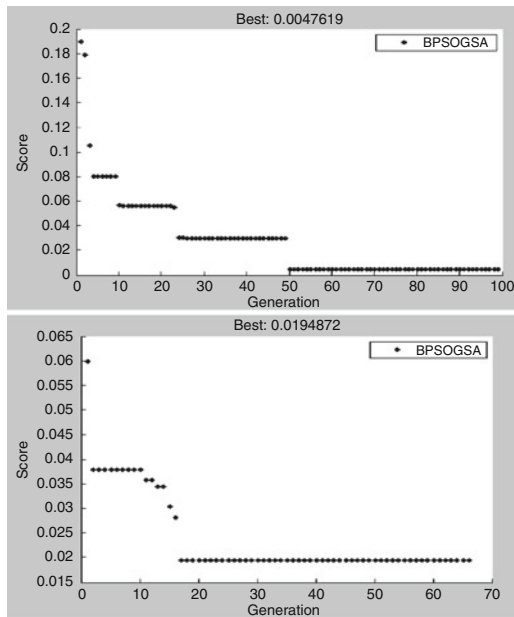


Fig. 26.5 Results obtained for lungcancer and parkinsons dataset for BPSOGSA

Table 26.4 The number features corresponding to best solutions

Data sets	No features	Best result
breast	9	1 4 8 15 18 21 22 28 29
congress	5	2 3 4 9 11
heart	7	3 5 9 10 11 12 13
ionosphere	9	2 5 9 13 14 15 17 24 27
krvskp	18	1 6 7 10 11 15 16 17 18 20 21 22 23 27 29 30 33 35
landsat	36	1 2 3 5 7 9 10 12 14 16 17 18 19 24 25 27 28 29 34 35
lungcancer	14	2 3 4 6 8 17 18 19 23 30 34 36 37 46
parkinsons	6	1 17 18 20 21 22
semeion	113	1 2 3 7 8 9 10 12 17 20 21 ...
sonar	60	4 6 7 9 10 13 15 16 20 22 25 34 36 40 41 45 46 48 53 60

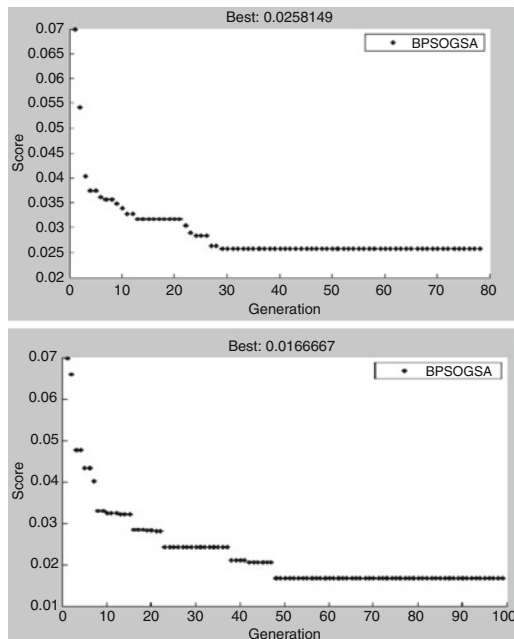


Fig. 26.6 Results obtained for semeion and sonar dataset for BMPSOGSA

all dataset. Even if BMPSOGSA outperforms GA in the five first dataset or gives the same results and converges rapidly than GA, GA performance is better than BMPSOGSA in the remaining dataset. This issue may be resolved by adding an adaptive mutation strategy to BMPSOGSA to enhance the mutation performance. We notice also from Table 26.3 that the local search has improved the accuracy of BPSOGSA in a number of datasets (5, 7, 8 and 10). However, the LS (pattern search) is much more computational demanding of resources than the mutation operator.

26.4.3 Comparison with Well-Known FS Techniques

To further examine the BMP SOGSA algorithm, we compare it with other well known wrapper FS techniques. The considered approaches are mutual information (MI), statistical dependency (SD), random subset feature selection (RSFS), sequential forward selection (SFS) and sequential floating forward selection (SFFS). There have been used in [22]. We have adapted the available code of these methods to the ten benchmark dataset. In this section, the fitness information is limited to the classification error using KNN classifier as in the mentioned paper because the number of the selected features depends on the methods parameters which are defined by the user. For instance, mutual information (MI) and statistical dependency (SD) consist of ranking variables and choosing the best ones. In this experiment, the number of chosen features is set to the third of the total number of features (if the third is not an integer, we use the nearest integer to it) (Table 26.5).

Table 26.5 Comparison of BMP SOGSA performance with well-known FS methods. Bold values are the best obtained results

Data sets		BMP SOGSA	SD	MI	RSFS	SFS	SFFS
breast	Error	0.0158	0.0510	0.0510	0.0475	0.0158	0.0756
	Features	9	10	10	10	10	2
congress	Error	0.02069	0.0391	0.0391	0.0.0391	0.0276	0.0414
	Features	5	5	5	3	5	2
heart	Error	0.085182	0.1630	0.1630	0.1556	0.1630	0.1630
	Features	7	4	4	5	4	4
ionosphere	Error	0.0254	0.1026	0.0826	0.0741	0.0313	0.0684
	Features	9	11	11	12	8	3
krvskp	Error	0.0160	0.0788	0.0788	0.1724	0.0335	0.3069
	Features	18	12	12	5	12	2
landsat	Error	0.0751	0.1803	0.1737	–	–	– ^a
	Features	21	12	12	–	–	– ^a
lungcancer	Error	0.0313	0.3125	0.2188	0.1875	0.1250	0.1250
	Features	17	19	19	8	4	4
parkinsons	Error	0.0051	0.1077	0.1077	0.0410	0.0205	0.1590
	Features	9	7	7	8	7	2
semeion	Error	0.0301	0.0722	0.0722	0.7589	0.0621	0.7533
	Features	115	85	85	4	42	2
sonar	Error	0.0192	0.026127	0.0433	0.1154	0.0048	0.4183
	Features	26	20	20	11	17	2

^a The problem cannot be solved by the used tools. This is probably caused by the huge number of iterations of RSFS, SFS and SFFS (the number of instances of this sample is 6435)

As a first remark, we can see that BMP SOGSA and SFS give better results than the other well known methods which are considered in this experiment. BMP SOGSA has better results than SD, MI, RSFS and SFFS. In more details, SD and MI give almost similar results which are lower than BMP SOGSA. Moreover, they are not adapted if the number of instances is less than the number of features (lung-cancer). We can see also that SFFS is more restricted in terms of number of features

(caused by the “floating” concept). We can see also that BMP SOGSA gives better results than SFS in most cases. Therefore, we can confirm again that the metaheuristics are in most cases more adapted to the FS problem than traditional methods and that our proposed BMP SOGSA algorithm is promising for FS.

26.5 Conclusion

In this paper, we have proposed a binary hybridization of GSA and PSO algorithms enhanced by a mutation operator. Our aim was to investigate the performance of this hybrid metaheuristic in wrapper feature selection methods. Experimental results show that the proposed BMP SOGSA has given better computational performance than BPSO, BGSA and BPSOGSA-Sig in all benchmark datasets and competitive results to GA. Moreover, it has outperformed a number of well-known methods for FS. However, the mutation operator can still be extended to clearly improve the BPSOGSA algorithm. Therefore, future research should attempt to add a learning strategy to the mutation operator in order to enhance its impact on the adaptivity of our algorithm.

References

1. Z. Beheshti, S.M. Shamsuddin, S. Hasan, Memetic binary particle swarm optimization for discrete optimization problems. *Inf. Sci.* **299**, 58–84 (2015)
2. A.R. Behjat, A. Mustapha, H. Nezamabadi, M.N. Sulaiman, N. Mustapha, Feature subset selection using binary gravitational search algorithm for intrusion detection system, in *Intelligent Information and Database Systems*. Lecture Notes in Computer Science, vol. 7803 (Springer, Berlin/Heidelberg, 2013), pp. 377–386
3. G. Brown, A. Pocock, M.-J. Zhao, M. Lujan, Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *J. Mach. Learn. Res.* **13**, 27–66 (2012)
4. T. Chakraborti, A. Chatterjee, A novel binary adaptive weight GSA based feature selection for face recognition using local gradient patterns, modified census transform and local binary patterns. *Eng. Appl. Artif. Intel.* **33**, 80–90 (2014)
5. R. Diao, Q. Shen, Nature inspired feature selection meta-heuristics. *Eng. Appl. Artif. Intel.* **44**(3), 311–340 (2015)
6. T. Ganesan, I.M. Elamvazuthi, K.Z. Ku Shaari, P. Vasant, Swarm intelligence and gravitational search algorithm for multi-objective optimization of synthesis gas production. *Appl. Energy* **103**, 368–374 (2013)
7. J. Garca-Nieto, E. Alba, L. Jourdan, E. Talbi, Sensitivity and specificity based multiobjective approach for feature selection: application to cancer diagnosis. *Inf. Process. Lett.* **109**, 887–896 (2009)
8. I. Guyon, A. Elisseeff, An introduction to variable and feature selection. *J. Mach. Learn. Res.* **3**, 1157–1182 (2003)
9. X.H. Han, X.M. Chang, L. Quan, X.Y. Xiong, J.X. Li, Z.X. Zhang, Y. Liu, Feature subset selection by gravitational search algorithm optimization. *Inf. Sci.* **281**, 128–146 (2014)

10. Z. Hu, Y. Bao, T. Xiong, Comprehensive learning particle swarm optimization based memetic algorithm for model selection in short-term load forecasting using support vector regression. *Appl. Soft Comput.* **25**, 15–25 (2014)
11. S. Jiang, A novel hybrid particle swarm optimization and gravitational search algorithm for solving economic emission load dispatch problems with various practical constraints. *Electr. Power Energy Syst.* **55**, 628–644 (2014)
12. L. Jourdan, C. Dhaenens, E.-G. Talbi, Evolutionary feature selection for bioinformatics, in *Computational Intelligence in Bioinformatics*, Chap. 6 (IEEE Press, New York, 2007), pp. 117–139
13. J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm optimization, in *Proceedings on Conference on Systems, Man and Cybernetics*, vol. 4 (1997), pp. 104–109
14. R. Kohavi, G.H. John, Wrappers for feature subset selection. *Artif. Intell.* **97**(1–2), 273–324 (1997)
15. J. Krause, J. Cordeiro, R.S. Parpinelli, H.S. Lopes, A survey of swarm algorithms applied to discrete optimization problems, in *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications* (Elsevier Science & Technology Books, London, 2013), pp. 169–191
16. J. Lee, D.-W. Kim, Memetic feature selection algorithm for multi-label classification. *Inf. Sci.* **293**, 80–96 (2015)
17. Y. Lu, I. Cohen, X.S. Zhou, Q. Tian, Feature selection using principal feature analysis, in *Proceedings of the 15th International Conference on Multimedia, MULTIMEDIA '07* (ACM, New York, 2007), pp. 301–304
18. S. Mallick, S.P. Ghoshal, P. Acharjee, S.S. Thakur, Optimal static state estimation using improved particle swarm optimization and gravitational search algorithm. *Electr. Power Energy Syst.* **52**, 254–265 (2013)
19. S. Mirjalili, G.-G. Wang, L.D. Coelho, Binary optimization using hybrid particle swarm optimization and gravitational search algorithm. *Neural Comput. Appl.* **25**(6), 1423–1435 (2014)
20. M. Nikravesh, L.A. Zadeh, I. Guyon, S. Gunn, *Feature Extraction, Foundations and Applications* (Springer, Berlin, 2006)
21. S. Piramuthu, Evaluating feature selection methods for learning in data mining applications. *Eur. J. Oper. Res.* **156**, 483–494 (2004)
22. J. Pohjalainen, O. Rsnen, S. Kadioglu, Feature selection methods and their combinations in high-dimensional classification of speaker likability, intelligibility and personality traits. *Comput. Speech Lang.* **29**, 145–171 (2015)
23. E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, BGSA: binary gravitational search algorithm. *Nat. Comput.* **9**(3), 727–745 (2010)
24. Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in *Proceedings of the IEEE World Congress of Computational Intelligence* (1997), pp. 69–73
25. E.-G. Talbi, L. Jourdan, J. Garcia-Nieto, E. Alba, Comparison of population based meta-heuristics for feature selection: application to microarray data classification, in *AICCSA'2008 IEEE/ACS International Conference on Computer Systems and Applications*, Doha, Nov 2008, pp. 45–52
26. A. Unler, A. Murat, A discrete particle swarm optimization method for feature selection in binary classification problems. *Eur. J. Oper. Res.* **206**, 528–539 (2010)
27. J.R. Vergara, P.A. Estvez, A review of feature selection methods based on mutual information. *Neural Comput. Appl.* **24**(1), 175–186 (2014)
28. L. Vignolo, D. Milone, J. Scharcanski, Feature selection for face recognition based on multi-objective evolutionary wrapper. *Expert Syst. Appl.* **40**, 5077–5084 (2013)
29. B. Xue, M. Zhang, W. Browne, Particle swarm optimisation for feature selection in classification: novel initialisation and updating mechanisms. *Appl. Soft Comput.* **18**, 261–176 (2014)
30. Y. Zhang, S. Wang, P. Phillips, G. Ji, Binary PSO with mutation operator for feature selection using decision tree applied to spam detection. *Knowl.-Based Syst.* **64**, 22–31 (2014)

Chapter 27

An Optimal Deployment of Readers for RFID Network Planning Using NSGA-II

Abdelkader Raghieb, Badr Abou El Majd, and Brahim Aghezzaf

Abstract Radio frequency identification (RFID) is an automated data collection technology with the aim to facilitate data acquisition and storage without human intervention. RFID process depends on radio-frequency waves to transfer data between a reader and an electronic tag attached to an item, in order to identify objects or persons, which allows an automated traceability. The deployment of RFID readers is an important component in RFID system, and plays a key role in RFID Network Planning (RNP). Therefore, in order to optimize the deployment of RFID reader problem, we propose a new approach based on multi-level strategy using as main objectives the coverage, the number of deployed readers and the interference. In this way, Non-dominated Sorting Genetic algorithm II (NSGA-II) is adopted in order to minimize the total quantity of readers required to identify all tags in a given area. The proposed multi-level approach based on NSGA-II algorithm has a several attractive features which makes it ideal for our research and the simulation results show its effectiveness and performance.

Keywords NSGA-II • RFID • RFID network planning • Deployment • Multi-objective problem • Optimization

27.1 Introduction

The term RFID refers to the radio frequency identification technology. It's an automated data collection technology using radio-frequency waves to transfer data between a reader and a tag which consist to identify, track and do management of material flow. Figure 27.1 presents the basic components of an RFID system.

A. Raghieb (✉) • B. Aghezzaf
LIMSAD Laboratory, Faculty of Sciences Ain Chock, Hassal II University of Casablanca, Casablanca, Morocco
e-mail: raghib.abdelkader@hotmail.com; b.aghezzaf@fsac.ma

B. Abou El Majd
Laboratory of Computer Science and Decision Support, Faculty of Sciences, Casablanca, Morocco
e-mail: b.abouelmajd@fsac.ac.ma

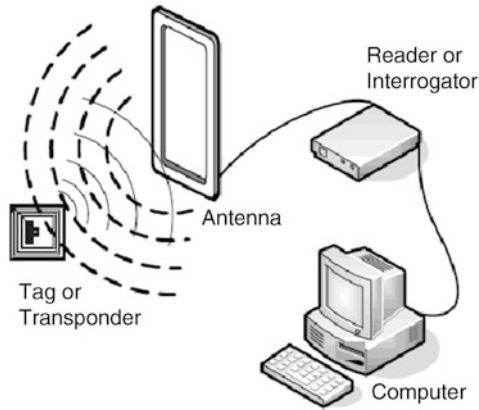


Fig. 27.1 Basic components of a typical RFID system

The identification of objects using RFID system depends on the communication process between a reader and an electronic tag, attached to an item. This process depends on correlated factors: the position and orientation of RFID tags, the interference from the environment, the type of material to identify and infrastructure limitations, etc.

One factor that is under control is the location and orientation of RFID reader. Generally, in order to detect and monitor the tags in RFID network, multiple readers are required according to their limited interrogation range. The cost and the number of tags covered directly depend on the number of deployed readers. Therefore, with the aim to minimize the deployment costs, there is a need to determine the optimal quantity and positions of RFID reader in the working area without losing efficiency in the communication process.

In the new global RFID system, the deployment of the readers has become a central issue for the RFID network planning. The aim is to optimize the deployment of RFID readers by covering all the tags in the entire region using a few number of readers. Therefore, several attempts have been made to achieve the optimal RFID network planning, mostly by using the multi-objective optimization [7] based on operators and algorithms. There is a large volume of published studies describing the role of the RFID Network Planning (RNP) and presenting several keys in order to achieve the optimal deployment of RFID readers. As in [9], authors presented a tentative reader elimination operator (TRE) based on PSO algorithm, and with the aim to delete and recover the deployed readers during the search process. Detailed

multi-objective optimization of RNP showed in [6, 2], in order to find the globally best Pareto-optimal solutions of the problem. In other major studies and with the aim to propose a new way to solve RNP problem [12, 4, 13], describe a new optimization algorithms under the name of hierarchical artificial bee colony (HABC), the multi-colony bacteria foraging optimization (MC-BFO), and the cooperative multi-objective artificial colony (CMOABC), respectively. Hsu and Yuan [10], Lu and Yu [11], and Chen et al. [5] provide in-depth analysis and results of the RNP problem, by using new models in order to obtain an optimal deployment.

The aim of this paper is to propose an efficient multi-level approach, based on the non-dominated sorting genetic algorithm II (NSGA-II), with several attractive features: determining the minimum number of readers, finding their optimal positions, guarantying a full coverage of all the tags and minimizing the interference of the environment, etc.

The overall structure of this paper takes the form of five sections. Section 27.2 begins by laying out the theoretical dimension of the deployment of RFID readers problem, and gives its mathematical formulation. Section 27.3 is concerned with the methodology and the methods used to solve the deployment problem. For that, we define the non-dominated sorting genetic algorithm II (NSGA-II). Also, we describe the two proposed approaches (the multi-level and the multi-objective optimization). Section 27.4 presents the numerical results. Finally, we conclude and give some perspectives.

27.2 Problem Formulation

In this section, we outline the different objectives and constraints that we considered for the optimal deployment of RFID readers. The goal is to find a set of the coordinates readers such that all the tags of the whole area is covered. At the same time, the total number of readers and their interference must be minimized. Figure 27.2 shows an example of RFID deployment: readers are presented by red cross and their interrogation range by red circles, tags are presented by blue diamond nodes.

Throughout this paper, we use the following notations:

- N_R : Number of the available RFID readers.
- N_T : Number of the tags deployed in the area.
- IR_j : Interrogation range of the j th reader.
- d_{ji} : Distance between the j th reader and the i th tag.

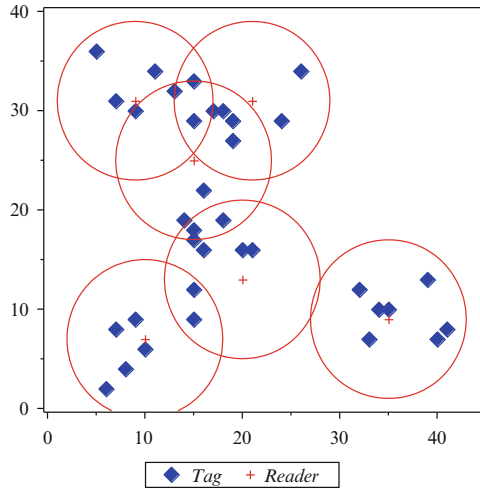


Fig. 27.2 Example of RFID deployment

27.2.1 Number of Deployed Readers

The first objective function represents the number of RFID readers, which is an important index for evaluating the performance of the deployment. Minimizing the total number of readers aims to reduce the cost. The optimization algorithm will search for the minimum value of this objective function. It can be defined as:

$$f_1 = \sum_{j=1}^{N_R} r_j$$

Where,

$$r_j = \begin{cases} 1 & \text{if the } j\text{th reader is deployed.} \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, the algorithm finds the optimal position of each deployed reader, and satisfies the constraint which ensures that we can only place each reader within a certain area. For a rectangular area with height H and width W , this constraint is satisfied:

$$0 \leq x_j \leq H \text{ and } 0 \leq y_j \leq W \quad \forall j = 1, \dots, N_R$$

27.2.2 Full Coverage

The main objective function for the deployment of RFID readers is to cover all the tags in the whole space. That means each tag must be in the interrogation range of

at least one deployed reader. Therefore, we should reduce the number of non-cover tags, which can be formulated as follows:

$$f_2 = N_T - \sum_{i=1}^{N_T} y_{ji} \quad (j = 1 \dots N_R)$$

Where,

$$y_{ji} = \begin{cases} 1 & \text{if } \exists! j = 1 \dots N_R \text{ such that } (d_{ji} \leq IR_j \text{ and } r_j = 1) \\ 0 & \text{otherwise.} \end{cases}$$

27.2.3 Interference

The interference is the interrogation of a tag with several readers at the same time. It decreases the efficiency of the RFID system. As we can see in Fig. 27.3, the interference mainly occurs when at least two deployed readers in the area cover a tag.

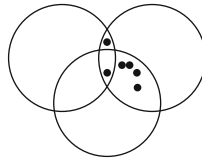


Fig. 27.3 Example of the interference of six tags covered by three RFID readers

The interference must be reduced by using the following expression:

$$f_3 = \sum_{i=1}^{N_T} c_i$$

Where,

$$c_i = \begin{cases} 1 & \text{if } \sum_{j=1}^{N_R} y_{ji} \geq 2 \\ 0 & \text{otherwise.} \end{cases}$$

27.3 Methods

In order to find an optimal deployment of the RFID readers, the NSGA-II algorithm is adopted for this investigation due to its several advantages.

27.3.1 NSGA-II Algorithm

The term NSGA-II has come to be used to refer to the meta-heuristic Non-dominated Sorting Genetic algorithm II [8], and can be defined as a recent multi-objective genetic algorithm.

The genetic algorithm (GA) is a famous population-based search algorithm which adopted in the NSGA-II algorithm. It was introduced originally by J. Holland in 1975, inspired on Darwin's principle of natural selection (Survival of the fittest). This optimization algorithm explores the search space of a problem by maintains a population of candidate solutions, and makes it evolve by iteratively applying a set of stochastic operators, in order to optimize one or several objectives.

The GA has a several attractive features: strong robustness, simplicity of the concepts, high efficiency, flexibility, parallelism, etc. This advantage makes GA more robust, and it has been successfully applied in many areas and solved a variety of optimization problems in a faster and cheaper way [3, 14, 15, 1].

Each chromosome of the population which represents a candidate solution of the problem explores the search space effectively, in order to obtain better results. Therefore, the GA's process begins by evaluating the fitness (performance) of the chromosomes. According on the evaluation's results, and by using simple evolutionary operators (selection, crossover and mutation), some individuals reproduce, others disappear and only the best adapted individuals are expected to survive in order to create a new efficiently population.

It is becoming increasingly difficult to ignore the high computational complexity, the non elitism approach and the need for specifying a sharing parameter when solving multi-modal problems by using multi objective evolutionary algorithms [8]. NSGA-II with three special characteristics, fast non-dominated sorting approach, fast crowded distance estimation procedure and simple crowded comparison operator can be used to limit the previous difficulties.

As we can see in Fig. 27.4 [8], the basic of the NSGA-II algorithm is very simple. It starts by initializing randomly a population, and sorting all the individual according to the non-domination criteria. Moreover, the individuals are selected based on the rank and the crowding distance in order to guarantee diversity and spread of solutions. Also, the solutions will be modified by using specific crossover and mutation operators with the aim to generate potentially better ones. Offspring and current generation population are combined and the individuals of the next generation are set by selection and repeats until achieving one of the stopping criterion.

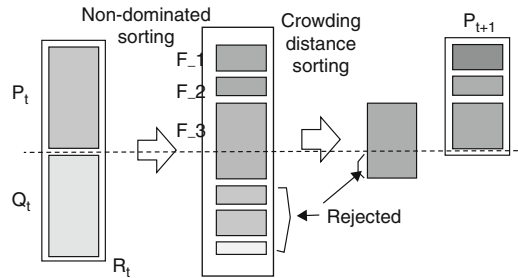


Fig. 27.4 Procedure of the non-dominated sorting genetic algorithm II (NSGA-II) at the t th generation

27.3.2 The Proposed Approaches

In order to find a full coverage of all the tags distributed in the search space and minimizing their interferences, by using a minimal number of the RFID readers and finding their optimal positions, we presented a mathematical model based on NSGA-II algorithm for solving the multi-objective RFID readers deployment problem.

$$\begin{cases} \text{Min } F(x, y, r) = (f_1(x, y, r), f_2(x, y, r), f_3(x, y, r)) \\ (x, y, r) \in (\mathbb{R}_+^N, \mathbb{R}_+^N, \{0, 1\}^N) \end{cases}$$

Each chromosome i of the population contains N available readers according to the decision maker.

$$(x_i^1, y_i^1, r_i^1, x_i^2, y_i^2, r_i^2, \dots, x_i^N, y_i^N, r_i^N)$$

where (x_i^k, y_i^k) is the coordinate of the k th reader and r_i^k its availability, which equals 1 if its deployed and 0 otherwise. N is the maximum number of RFID readers that can be deployed in the working area.

Our approach transforms the multi-objective problem, explained above, into an equivalent bi-objective problem. For that we presented a mathematical model for obtaining robust solutions using the multi-level approach. The first objective, associated to the number of deployed readers, is included in the multi-level concept.

$$\begin{cases} \text{Min } F(x, y) = (f_2(x, y), f_3(x, y)) \\ (x, y) \in (\mathbb{R}_+^L, \mathbb{R}_+^L) \end{cases}$$

where L is the level of the proposed approach as shown in Table 27.1. Moreover, it's the number of deployed readers, and the dimension of each individual of the population.

Each chromosome i contains one or more readers according to the dimension of the NSGA-II and without fixing the number of available readers, and each one of them is represented by its coordinate.

$$(x_i^1, y_i^1, x_i^2, y_i^2, \dots, x_i^N, y_i^N)$$

where (x_i^k, y_i^k) is the coordinate of the k th reader. The k th level means that we use the NSGA-II algorithm to find the best solution which covers the maximum of tags by using k readers in each chromosome.

Table 27.1 Representation of each chromosome of the population

Levels	i th chromosome of the population
Level 1	(x_i^1, y_i^1)
Level 2	$(x_i^1, y_i^1, x_i^2, y_i^2)$
.	.
.	.
.	.
Level N	$(x_i^1, y_i^1, x_i^2, y_i^2, \dots, x_i^N, y_i^N)$

As we can see in Fig. 27.5, the proposed multi-level approach starts by deploying randomly a reader, and then updates its position in order to determine the best location required to cover the maximum number of tags. At each update, the best position is searched until we reach the full coverage, or we increment the number of readers when a search timeout occurs.

The proposed approach has a several of attractive features:

- Finding an optimal deployment without fixing the number of available readers in the beginning.
- Having complete reports of all levels in order to have many optimal solutions and facilitate the decision aiding to determine the best judgment of all the scenarios.
- Transforming the multi-objective deployment problem to a bi-objective problem, which reduce the complexity.
- Minimizing data quantity by controlling the chromosome's dimension and the number of readers in all levels.
- Reducing the number of decision variables, which decrease the chromosome's length.

27.4 Results and Discussion

The experiments were conducted to evaluate and validate the performance of the proposed approach. In this case, we considered six benchmarking problems namely C30, R30, C50, R50, C100 and R100, which contain 30, 50 and 100 tags deployed in the working area. All the six problems have been described in [9]. The readers and the tags are distributed in an area of 50*50 m.

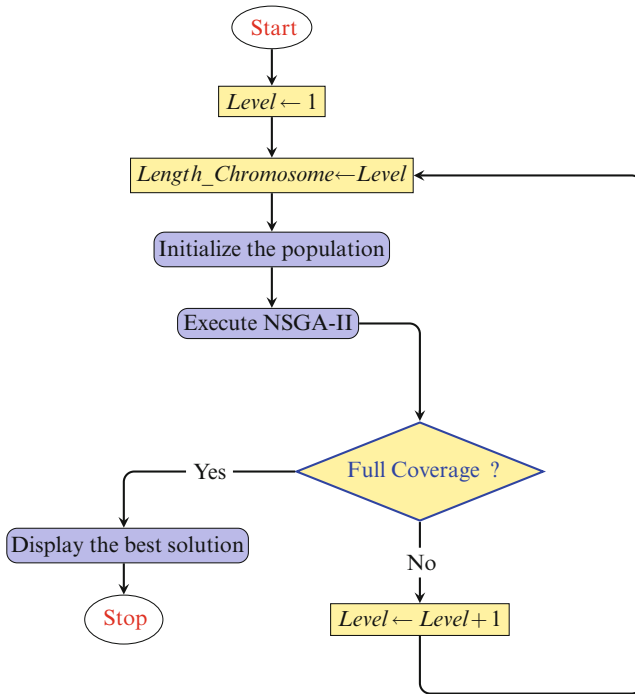


Fig. 27.5 Flowchart of the proposed Multi-Level approach

The comparisons between the two approaches (the multi-level and the multi objective optimization) were made using a set of parameter settings as shown in Table 27.2.

Table 27.2 Properties of multi-objective and multi-level optimization by using NSGA-II algorithm

Parameter	Value	
	Multi objective	Multi level
Area height H	50.0	50.0
Area width W	50.0	50.0
Number of available readers	N	–
Maximum iterations	25,000	10,000
Population size	20	20
Length of chromosome	N	Level dimension
Number of run	10	10

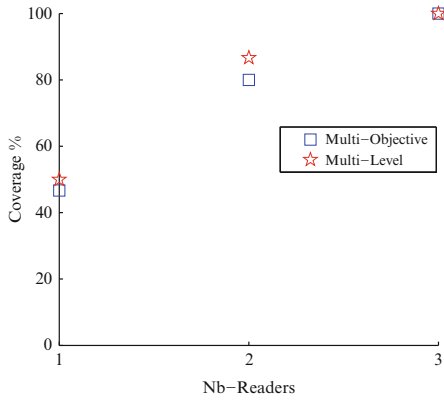


Fig. 27.6 C30's Pareto-front obtained by using Multi-Level and Multi-Objective Optimization

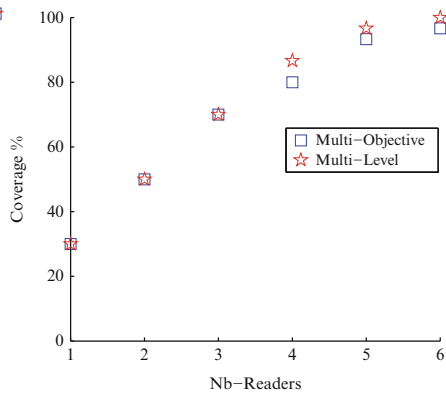


Fig. 27.7 R30's Pareto-front obtained by using Multi-Level and Multi-Objective Optimization

Table 27.3 The results of the two approaches for solving the six benchmarks

Benchmarks		C30	C50	C100	R30	R50	R100	
Multi objective	Best	Coverage	100%	100%	100%	100%	100%	
		Readers	3	5	5	6	8	8
		Interference	0	0	0	0	0	2
	Mean	Coverage	100%	100%	100%	98%	98.4%	98.4%
		Readers	3	5	5	6.2	7.4	7.7
		Interference	0	0	0	0.4	0.8	5
	Worst	Coverage	100%	100%	100%	100%	100%	97%
		Readers	3	5	5	7	8	8
		Interference	0	0	0	2	1	5
Multi level	Best	Coverage	100%	100%	100%	100%	100%	100%
		Readers	3	5	5	6	7	8
		Interference	0	0	0	0	0	0
	Mean	Coverage	100%	100%	100%	100%	100%	100%
		Readers	3	5	5	6.8	7.7	8
		Interference	0	0	0	0	0	1.6
	Worst	Coverage	100%	100%	100%	100%	100%	100%
		Readers	3	5	5	7	8	8
		Interference	0	0	0	0	0	4

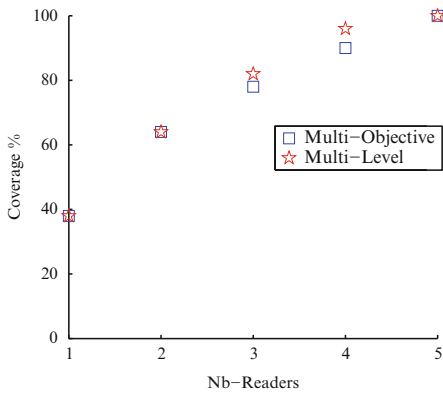


Fig. 27.8 C50's Pareto-front obtained by using Multi-Level and Multi-Objective Optimization

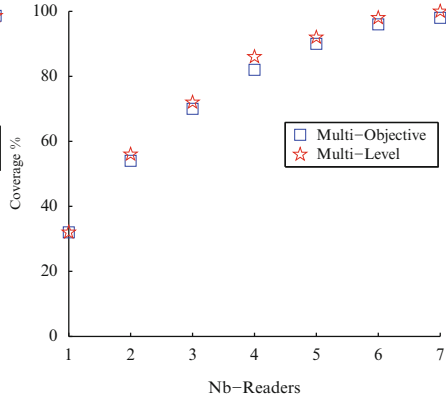


Fig. 27.9 R50's Pareto-front obtained by using Multi-Level and Multi-Objective Optimization

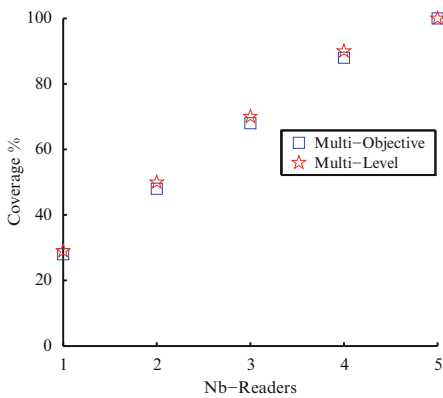


Fig. 27.10 C100's Pareto-front obtained by using Multi-Level and Multi-Objective Optimization

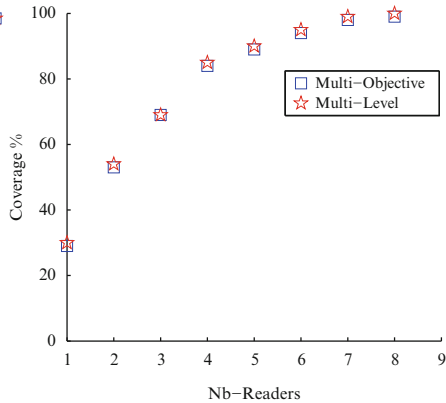


Fig. 27.11 R100's Pareto-front obtained by using Multi-Level and Multi-Objective Optimization

Table 27.3 presents the global results obtained from all the six problems by using the multi-level and the multi-objective optimization based on NSGA-II algorithm. It can be seen clearly that our proposed approach outperforms the multi-objective approach even in the worst runs. On all the six problems, it achieves the full coverage with the optimal number of deployed readers in almost all the run even for the complex problems (R30, R50 and R100).

These results are graphically supported in Figs. 27.6, 27.7, 27.8, 27.9, 27.10, and 27.11 by a set of Pareto optimal solution for the six problems according to two objectives (the tags coverage and the number of deployed readers). It's clearly

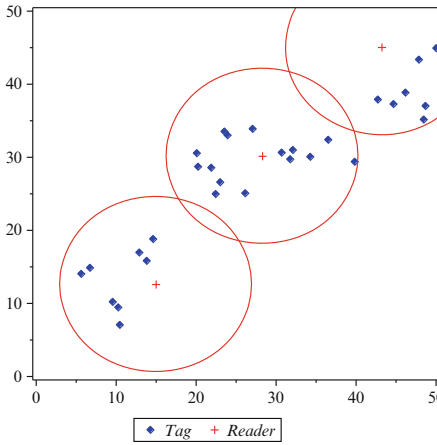


Fig. 27.12 The optimal deployment of C30's problem

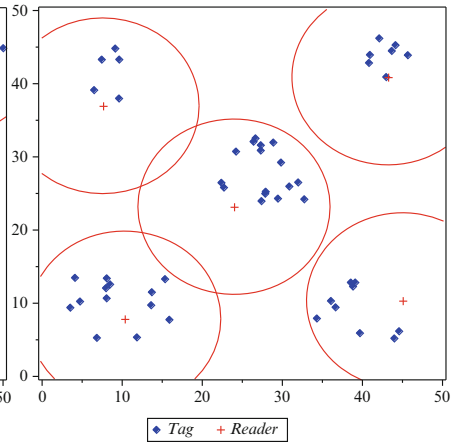


Fig. 27.13 The optimal deployment of C50's problem

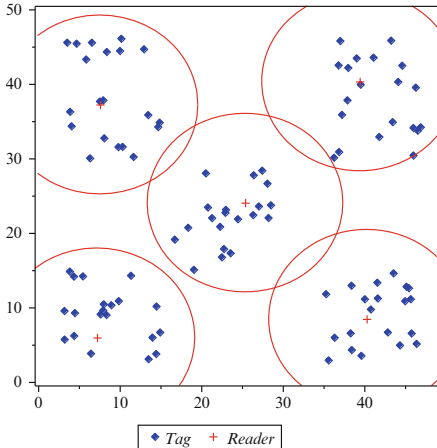


Fig. 27.14 The optimal deployment of C100's problem

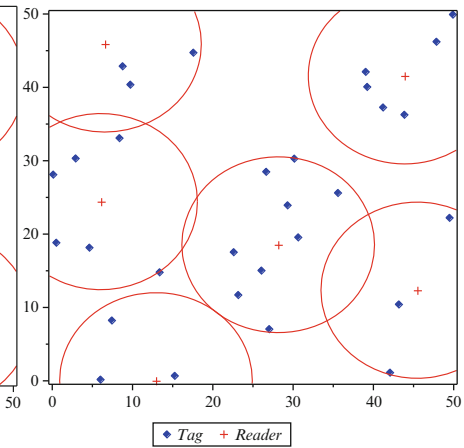


Fig. 27.15 The optimal deployment of R30's problem

observed and confirmed that the multi-level approach as the robust strategy providing the best overall performance by guarantying the quality of the non dominated solutions than the multi-objective optimization and yield higher accuracy and efficiency in a faster and cheaper way. Also, as we can see in the Pareto's figures, our approach obtains always the optimal solutions, which facilitate the decision aiding by giving a global vision of all the scenarios in order to increase the quality of the RFID system. All the optimal deployment of RFID readers for the six benchmark problems are shown in Figs. 27.12, 27.13, 27.14, 27.15, 27.16, and 27.17.

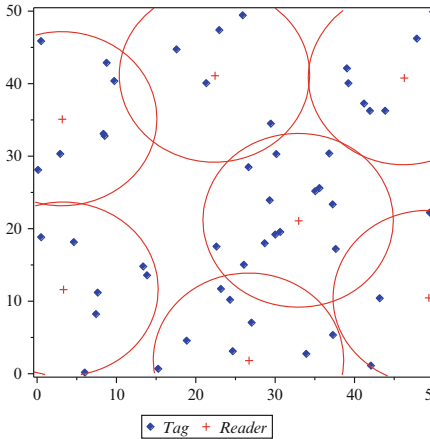


Fig. 27.16 The optimal deployment of R50's problem

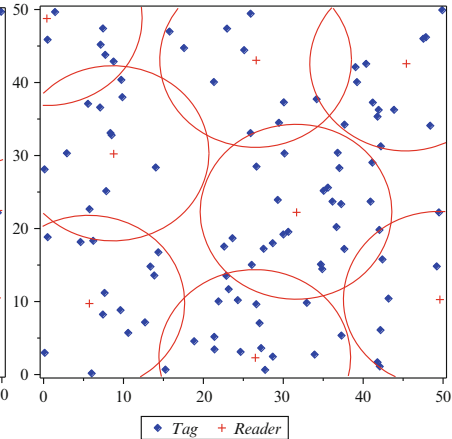


Fig. 27.17 The optimal deployment of R100's problem

27.5 Conclusion

This study set out to determine an optimal solution of deployment readers problem for the multi objective RFID network planning. The results obtained demonstrate clearly the efficiency and the robustness of the proposed approach by determining a minimal number of readers, finding their optimal positions, guaranteeing a full coverage of all tags in the working area and minimizing their interferences. As perspectives, we plan to apply the multi-level approach in the airport, with the aim to achieve an optimal deployment of RFID readers for tracking the luggage, which is a part of a global contribution of a complete RFID system application. Moreover, we suggest to apply the proposed approach for more complex benchmarks and real applications.

References

1. J.M. Arroyo, F.J. Fernández, Application of a genetic algorithm to n-K power system security assessment. *Int. J. Electr. Power Energy Syst.* **49**, 114–121 (2013). doi:10.1016/j.ijepes.2012.12.011
2. O. Botero, H. Chaouchi, RFID network topology design based on Genetic Algorithms, in *2011 IEEE International Conference on RFID-Technologies and Applications (RFID-TA)* (2011), pp. 300–305
3. T. Brodmeier, E. Pretsch, Application of genetic algorithms in molecular modeling. *J. Comput. Chem.* **15**, 588–595 (1994). doi:10.1002/jcc.540150604
4. H. Chen, Y. Zhu, K. Hu, Multi-colony bacteria foraging optimization with cell-to-cell communication for RFID network planning. *Appl. Soft Comput.* **10**, 539–547 (2010). doi:10.1016/j.asoc.2009.08.023

5. H. Chen, Y. Zhu, K. Hu, T. Ku, RFID network planning using a multi-swarm optimizer. *J. Netw. Comput. Appl.* **34**, 888–901 (2011). doi:10.1016/j.jnca.2010.04.004
6. H. Chen, Y. Zhu, L. Ma, B. Niu, Multiobjective RFID network optimization using multiobjective evolutionary and swarm intelligence approaches. *Math. Probl. Eng.* **2014**, e961412 (2014). doi:10.1155/2014/961412
7. K. Deb, D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms* (Wiley, New York, NY, 2001)
8. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**, 182–197 (2002). doi:10.1109/4235.996017
9. Y.-J. Gong, M. Shen, J. Zhang, et al., Optimizing RFID network planning by using a particle swarm optimization algorithm with redundant reader elimination. *IEEE Trans. Ind. Inf.* **8**, 900–912 (2012). doi:10.1109/TII.2012.2205390
10. C.-C. Hsu, P.-C. Yuan, The design and implementation of an intelligent deployment system for RFID readers. *Expert Syst. Appl.* **38**, 10506–10517 (2011). doi:10.1016/j.eswa.2011.02.109
11. S. Lu, S. Yu, A fuzzy k-coverage approach for RFID network planning using plant growth simulation algorithm. *J. Netw. Comput. Appl.* **39**, 280–291 (2014). doi:10.1016/j.jnca.2013.07.015
12. L. Ma, H. Chen, K. Hu, et al., Hierarchical artificial bee colony algorithm for RFID network planning optimization, hierarchical artificial bee colony algorithm for RFID network planning optimization. *Sci. World J.* **2014**, e941532 (2014). doi:10.1155/2014/941532
13. L. Ma, K. Hu, Y. Zhu, H. Chen, Cooperative artificial bee colony algorithm for multi-objective RFID network planning. *J. Netw. Comput. Appl.* **42**, 143–162 (2014). doi:10.1016/j.jnca.2014.02.012
14. A.P. McCabe, Constrained optimization of the shape of a wave energy collector by genetic algorithm. *Renew. Energy* **51**, 274–284 (2013). doi:10.1016/j.renene.2012.09.054
15. S. Wang, M. Liu, A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem. *Comput. Oper. Res.* **40**, 1064–1075 (2013). doi:10.1016/j.cor.2012.10.015

Chapter 28

An Enhanced Bat Echolocation Approach for Security Audit Trails Analysis Using Manhattan Distance

Wassila Guendouzi and Abdelmadjid Boukra

Abstract Security Audit Trail Analysis problem consists in detecting predefined attack scenarios in the audit trails. Each attack scenario is defined by a number of occurrences of auditable events. This problem is classified as an NP-Hard combinatorial optimization problem. In this paper, we propose to use the Bat echolocation approach to solve such problem. The proposed approach named an Enhanced Binary Bat Algorithm (EBBA) is an improvement of Bat Algorithm (BA). The fitness function is defined as the global attacks risks. In order to improve, the fitness function is combined with the Manhattan distance measure. Thus, intrusion detection process is guided, on one hand, by the fitness function that aims to maximize the global attacks risks and, on the other hand, by the Manhattan distance that attempts to reduce false Positives and false negatives. The best solution retained has the smallest Manhattan distance. Experiments show that the use of the Manhattan distance improves substantially the intrusion detection quality. The comparative study proves the effectiveness of the proposed approach to make correct prediction.

Keywords Intrusion detection • Security audit trail analysis • Combinatorial optimization problem • NP-Hard • Manhattan distance • Metaheuristics • Bat algorithm

28.1 Introduction

In the last years, the ubiquity of network and computer systems in modern society makes their protection a challenging task. Hackers and intruders, usually, exploit security vulnerabilities in order to compromise the integrity, availability and confidentiality of a computer resource. That is, traditional intrusion prevention alone like

W. Guendouzi • A. Boukra

Faculty of Electronics and Computer Science Laboratory LSI, USTHB BP 32 16111 El Alia, Bab-Ezzouar Algiers, Algeria

e-mail: wguendouzi@usthb.dz; aboukra@usthb.dz

© Springer International Publishing AG 2018

L. Amodeo et al. (eds.), *Recent Developments in Metaheuristics*,
Operations Research/Computer Science Interfaces Series 62,
DOI 10.1007/978-3-319-58253-5_28

477

firewall, authentication and data encryption mechanism have failed to completely provide protection against new and sophisticated types of intrusion. Therefore, a second line of defense is needed. Intrusion detection system (IDS) has become an important component in security infrastructures that aims to detect all intrusions in an efficient manner. Intrusion detection is the process of monitoring and analyzing security activities occurring in a computer or network systems [14].

IDSs can be categorized into different classes according to five concepts: usage frequency (real-time detection or batch detection), audit source location (Host IDS, Network IDS or Hybrid IDS), architecture (centralized or distributed), detection method (anomaly based or misuse based detection) and detection response (passive or active) [14]. The detection method is the brain of IDS.

In this paper, we are interested in misuse detection. The misuse mechanism aims to detect predefined attack scenarios in the audit trails. Each attack scenario is defined by a number of occurrences of auditable events. The temporal order of events sequences is not considered. The problem formulated in Sect. 28.3 is an NP-Hard combinatorial optimization problem [12]. Accordingly, it requires heuristic methods as databases of events and attacks grow. We investigate the efficiency of Bat Algorithm (BA) to solve such problem. BA is a powerful metaheuristic based on the echolocation principal of bats species when they navigate or chase prey. Our approach is an Enhanced Binary Bat Algorithm (EBBA) that combines fitness function with Manhattan distance to find the near optimal solution.

This paper is organized as follows. Section 28.2 presents the related work. Section 28.3 describes the problem formulation. Section 28.4 is dedicated to an overview of BA. Section 28.5 explains the proposed approach. Section 28.6 shows the experimental results and performance evaluation. Finally, we conclude in Sect. 28.7.

28.2 Related Work

In literature, different techniques, from different disciplines were used to develop efficient intrusion detection systems. The detection method can perform either anomaly based or misuse based detection. These two approaches are complementary. Some works propose to hybridize them [15].

In anomaly based approach, intrusions are identified as deviations from normal behavior. Statistical methods for anomaly detection were the first proposed in the area of intrusion detection. Other works based on learning techniques use neural network [16], immune systems [8] and genetic algorithm [11]. Clustering techniques are also employed [17].

Misuse detection approach consists of comparing recorded audit trail against predefined signatures. Generally, these techniques analyze audit trail files where are collected a set of user activities recorded from computer systems. Among the used techniques for misuse detection we can find genetic algorithm [1] and neural net-

work [2]. In [10] Lee uses rule association and frequent episode techniques. Swarm intelligence [9] is also applied to the intrusion detection field. Dass [6] and Mé [12] both use genetic algorithm to solve such problem. These two works were improved in [7] with new defined fitness functions and in [3, 4] and [5] using new Metaheuristics, BBO and HS respectively.

28.3 Problem Formulation

Audit trail analysis problem consists in searching predefined attack scenarios in the audit trails. Attack scenario is defined as a set of activities that intruders undertake on the computer systems. The proposed work is based on multiple fault diagnosis approach. This approach is analogous to the process of disease diagnosis in medicine. It aims to determine the set of conditions that may explain the presence of observed symptoms (the recorded events in the audit trails), using a specific knowledge of cause and effect (the attack scenarios).

This approach uses a predefined matrix of attack scenarios in which the temporal order of events sequences is not considered. Each attack scenario may be described as a set of couple (E_i, N_i) such that E_i is the event of type i and N_i is the occurrences number of this event in the audit trails. The mathematical formulation of the audit trail analysis problem is described by the formulas (28.1) and (28.2) :

$$\text{Max} \sum_{j=1}^{Na} R_j \times H_j \quad (28.1)$$

$$(AE \times H)_i \leq O_i, \quad 1 \leq i \leq Ne \quad (28.2)$$

Where:

- Ne is the number of events type.
- Na is the number of predefined attack scenarios.
- AE is the Attack/Event matrix of dimension $(Ne \times Na)$ that defines, for each attack, the events it generates. Each element $AE_{ij} \geq 0$ is the number of events of type i generated by the attack j .
- O is a vector of dimension Ne where each element O_i is the number of events of type i in the audit file. O is created from the analyzed audit file by counting for each event type the number of its occurrences in this file.
- R is a vector of dimension Na where $R_j > 0$ is the incurred risk of attack j .
- H is the hypothesis vector of dimension Na where $H_j = 1$ if attack j is present and $H_j = 0$ otherwise.

In the audit trail analysis, we aim to obtain the H vector that maximizes the Product $R \times H$ given in (28.1), with the constraint (28.2). The problem is to find the sub set of present attacks in the audit file. This problem is an NP-Hard combinatorial

optimization problem [12]. Thus, the application of exact algorithm is impossible for large size instances. The use of heuristics gains recognition. We opt in this paper to use the Bat echolocation principle to solve this problem.

28.4 Bat Algorithm Overview

BA mimics the echolocation principle of bats. It was introduced, firstly, by Yang [18]. Bats send out ultrasonic signals and examine returning echoes to analyze its surroundings. In BA, Yang has modeled the echolocation characteristics as follows [18]: Each artificial bat i flies randomly with frequency f_i , velocity v_i at position x_i . These three parameters are updated according to the formulas (28.3), (28.4) and (28.5) respectively [18]:

$$f_i = f_{min} + (f_{max} - f_{min}) \cdot \beta \quad (28.3)$$

$$v_i(t+1) = v_i(t) + (x_i(t) - Gbest) \cdot f_i \quad (28.4)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (28.5)$$

Where β is a random number of a uniform distribution within the interval $[0, 1]$, $Gbest$ is the best solution found until now and f_i is the frequency of i -th bat. The range of frequency is, usually, in the interval $[f_{min}, f_{max}]$.

BA performs a local search by generating a new local solution using local random walk given by formula (28.6) [18]:

$$x_{new} = x_{old} + \varepsilon A(t) \quad (28.6)$$

Where ε is a random number in $[-1, 1]$. ε attempts to control direction and strength of the bat random walk. $A(t)$ is the average loudness of all the bats at time t .

By analogy to the bats echolocation principle the loudness A_i and the rate r_i of pulse emission have to be updated. Bats can automatically adjust the loudness A_i and the rate r_i , depending on the proximity of their target. The loudness usually decreases and the rate of pulse emission increases. These two variables are updated as in formulas (28.7) (28.8) [18]:

$$A_i(t+1) = \alpha A_i(t) \quad (28.7)$$

$$r_i(t+1) = r_{max} [1 - \exp(-\gamma t)] \quad (28.8)$$

Where α and γ are constants fixed experimentally. Initially, each bat should has different values of loudness and pulse emission rate. These values are randomly chosen. At the final step of the algorithm, A_i will equal zero, while the final value of r_i is r_{max} .

A_i and r_i are updated only if the new solutions are enhanced to guarantee that bats are moving towards the near optimal solution. The basic steps of BA are summarized in [19].

28.5 Proposed Approach

In this section we describe the main components of the proposed approach named Enhanced Binary Bat Algorithm (EBBA). It is a balanced combination of exploitation and exploration, controlled by the echolocation parameters A_i and r_i . The general steps are depicted in Algorithm 1. Note that in what follows we will use indifferently H and x_i to signify the solution.

Algorithm 1 EBBA

```

Initialize the bat population :
     $X_i (i = 1, 2, \dots, p) = \text{rand} (0 \text{ or } 1)$  and  $V_i = 0$ 
    Initialise pulse rates  $r_i$  and the loudness  $A_i$ 
Fitness function  $f(x)$ ,  $x = (x_1, x_2, \dots, x_p)$ 
Define Pulse Frequency  $F_i$ ,  $\alpha$  and  $\gamma$ 
while  $t < \text{Max number of iteration}$  and there is no Solution with Distance Manhattan = 0 do
    for each bat  $b_i (i = 1..p)$  do
        if  $\text{rand}[0, 1] > r_i$  then
            if  $x_i$  is not feasible then
                Apply solution transformation (transform not feasible solution to feasible one)
            end if
            Apply Vertical permutation (Add intrusion to  $x_i$  by Transforming 0 Bits to 1)
            Apply Horizontal permutation (Rearrange the position of Bits equal to 1)
        end if
        if  $\text{rand}[0, 1] < A_i$  and  $f(x_i) \geq f(Pbest_i)$  then
            Accept the new solution
            Increase  $r_i$  and reduce  $A_i$ 
        end if
        Update  $Pbest_i$  and  $Globalbest$ 
        Update Global best distance (save the solution having the minimum Manhattan distance found until now)
        for each dimension  $j (j = 1..Na)$  do
            Adjusting frequency and updating velocities using equations (28.3) (28.4)
            Calculate transfer function value using equation (28.13)
            if  $\text{rand}[0, 1] < \text{transfer function value}$  then
                 $x_i^j = 0$ 
                Apply Horizontal permutation
            end if
        end for
    end for
end while

```

28.5.1 Solution Representation

The encoding for potential solutions is a binary vector H of dimension Na , where each element H_j is equal to 1 if the attack j is present and equal to 0 otherwise. The use of EBBA requires the simulation of virtual bats characterized each one by:

- Its position x_i that corresponds to the solution H . consequently, the search space is modeled as a discrete binary space of dimension Na . In each dimension j , the bat i can move between the positions x_i^j equal to 0 or 1.
- Its velocity v_i which is used to calculate the next bat position. In each dimension j , positions vary according to their own velocities v_i^j . Indeed, v_i is a vector of dimension Na in which each element v_i^j is updated using Eq. (28.4). The frequency f_i varies in the interval $[0,1]$.
- Its echolocation parameters: the intensity (A_i) and the pulse emission rate (r_i). We choose to vary these two parameters in the interval $[0, 1]$.

28.5.2 Initialization of Algorithm Parameters and Bat Population

The first step in the algorithm is to initialize the parameters α , γ , generation number (G) and the population size (P). We fix these parameters experimentally. Selected values are given in the next section.

The initial population is randomly generated. For each bat, the position vector x_i is a random combination of 0 and 1 and the velocity vector v_i is initialized to 0. According to the principle of BA, emission pulse rate (r_i) is initially small and thus must takes a value close to 0, however the intensity (A_i) is initialized to a value close to 1. At the final step of the algorithm, r_i tends to r_{max} (r_{max} equals to 1) and A_i tends to 0.

28.5.3 Fitness Function and Manhattan Distance

28.5.3.1 Fitness Function

During the execution of the algorithm, new solutions are generated from existing ones. Formula (28.9) defines the selective function.

$$Max \sum_{j=1}^{Na} R_j \times H_j \quad (28.9)$$

Under the constraint : $(AE \times H)_i \leq O_i, \quad 1 \leq i \leq Ne.$

A solution is feasible if it satisfies the defined constraint. That is, the numbers of the occurrence of the different events generated by all present attacks are less

than or equal to the number of events recorded in the observed vector (O). Because many solutions are not feasible (do not satisfy the constraint), we opt to transform not feasible solutions into feasible ones in order to reduce the execution time. The pseudo code of this method is given in Sect. 28.5.5.

28.5.3.2 Manhattan Distance

Experiments show that the use of BA with the defined fitness function (28.9) alone have failed to find the balance between: the need for detecting all possible attacks and the need for avoiding false positive (detection of attacks that do not exist). We have introduced the Manhattan distance combined with the fitness function in order to improve the intrusion detection.

Definition 28.1. The Manhattan distance between two points A and B, with the respective coordinates (X_A, Y_A) and (X_B, Y_B) is defined by (28.10):

$$d(A, B) = |X_B - X_A| + |Y_B - Y_A| \quad (28.10)$$

We define the Manhattan distance associated with the feasible solution as the Manhattan distance between the vector Prod defined in (28.12) and the vector O. this distance is given by the following formula:

$$d(prod, O) = \sum_{i=1}^{Ne} (O_i - prod_i) \quad (28.11)$$

Such that Prod is a vector of dimension Ne where each element $prod_i$ is the sum of the number of occurrences of event i generated by all the attacks of the solution H. $prod_i$ is defined by the following formula:

$$prod_i = \sum_{j=1}^{Na} (AE_{ij} \times H_j) \quad (28.12)$$

To illustrate this, consider the example depicted in Fig. 28.1. The Manhattan distance associated with the solution vector H is calculated as follows: $d(prod, O) = (3 + 0 + 2 + 4 + 2 + 2 + 1 + 0 + 59 + 35) = 108$

		ATTACK										O	Prod	O - Prod
		0	1	2	3	4	5	6	7	8	9			
EVENT	0	3										3	0	3
	1				1	1						1	1	0
	2				1			2			5	5	3	2
	3		3									4	0	4
	4			3		5						5	3	2
	5		5								3	5	3	2
	6			5		5				1		7	6	1
	7						1				1	1	1	0
	8	50					10	1				70	11	59
	9										30	35	0	35

H	0	0	1	1	0	1	1	0	1	0
---	---	---	---	---	---	---	---	---	---	---

Fig. 28.1 Example illustrating an instance of all structures manipulated by the algorithm to calculate the fitness value and the Manhattan distance associated with a solution

28.5.4 Proposed Discretisation

The problem of our interest is of discrete nature. Therefore, since standard BA is a continuous optimization algorithm, it cannot be used to solve such problem directly. In this section, we propose an adaptation of the binary version of BA to the problem. The main idea is to change the position of a virtual Bat according to the probability of its velocity [13]. We have opted for a v-shaped transfer function, defined by Eq. (28.13).

$$V(v_i^j(t)) = \left| \frac{2}{\pi} \arctan\left(\frac{\pi}{2} v_{ij}(t)\right) \right| \tag{28.13}$$

For a large absolute value of the velocity, the transfer function V-shaped returns a high probability of changing the position and it returns a small probability for a small absolute value of the velocity [13]. Accordingly, the position vector is updated using horizontal permutation described in the next subsection, as shown in the following rule :

if $V(v_i^j(t)) > \sigma$ **then**
 $x_i^j = 0$
 Apply Horizontal Permutation;
end if

Where σ is a random number in $[0,1]$. x_i^j and v_i^j indicate the position and velocity of i-th Bat in j-th dimension respectively. As shown in the rule above, for each dimension j, bats far from the global best solution (i.e. with a large absolute value of the velocity v_i^j) will have a high probability to change their position. In this case,

the detected intrusion j (x_i^j equal to 1) is ignored by updating its value to 0. Then horizontal permutation is applied to make deeper changes in the entire vector H . The details of this procedure are given in the next subsection. Notice that we opt to stop the Algorithm when it reaches the maximum of generation or when the Manhattan distance is equal to 0.

28.5.5 Operators

In this section, we present the main operators used in Algorithm 1, namely vertical permutation, horizontal permutation and solution transformation. The local search (exploitation) uses the Vertical permutation and horizontal permutation. Exploration uses only the horizontal permutation. Solution transformation is used to transform not feasible solution to feasible one. For vertical and horizontal operators, each time we modify the solution vector H , the Manhattan distance associated with it should be updated. The naive way is to compute it from the scratch. Nevertheless, it can take much time as data bases of events and attacks grow. For this, we propose a fast incremental calculation procedure in which the Prod vector associated with the current solution is used. This latter is updated and saved each time we add an intrusion into the current solution. In what follows, we explain these three operators and present the corresponding algorithms.

28.5.5.1 Solution Transformation

The transformation is performed by removing present intrusions that cause constraint violation. That is, changing these specific bats positions x_i^j from 1 to 0. Operator 1 describes this process.

Operator 1 Solution transformation procedure

```

for each row  $i$  of the AE matrix ( $i = 1..Ne$ ) do
   $Sous = O[i]$  //initialize the variable Sous
  //for each position in H
  for  $j=1$  to  $Na$  do
    if  $H[j] = 1$  and  $AE[i][j] = 0$  then
       $sous = sous - AE[i][j]$ 
    end if
    if  $Sous < 0$  then
       $H[j] = 0$  //Delete the intrusion  $j$ , because it caused a constraint violation (Sous is negative)
       $Sous = sous + AE[i][j]$  // reset the variable  $Sous$  to its former value
    end if
  end for
end for

```

28.5.5.2 Vertical Permutation

Vertical permutation consists in adding intrusions into the vector H without constraint violation. It is performed by changing a specific bats positions x_i^j from 0 to 1. The choice of the positions that will be changed is guided by the Manhattan distance. Positions x_i^j equal to 0 is converted to 1 in the case where the Manhattan distance associated with the initial vector H decreases. This operation is repeated until there is no Positions x_i^j equal to 0 in which their transformation to 1 decreases the Manhattan distance associated with H. The pseudo-code of the vertical permutation is given in Operator 2.

Operator 2 Vertical permutation procedure

```

Calculate the vector named distance, using the initial vector H
Distanceinit = Manhattan distance associated with the initial vector H
while H is updated do
    Distmin = minimum distance in the vector Distance
    Indicemin = index of Distmin in the vector Distance
    if Distmin < distanceinit then
        H[Indicemin] = 1 //add the intrusion
        Distance [Indicemin] = -1 // Update the distance element associated with the intrusion of
        Indicemin
        Update Distanceinit //Manhattan distance associated with the updated vector H (with H
        [Indicemin] = 1)
        Update the vector Distance using the updated vector H
    end if
end while

```

Such as *Distance* is a vector of dimension Na associated with a solution vector H. It is used to save the Manhattan distance after changing each positions from zero to one. Each element $Distance[i]$ can have one of the following values:

- $Distance[i]$ equals to -1 if the corresponding intrusion already exists ($H[i] = 1$).
- $Distance[i]$ equals to -1 if the corresponding intrusion do not exist ($H[i] = 0$) and changing it from 0 to one causes constraint violation.
- $Distance[i]$ equals to the Manhattan distance associated with H after changing the corresponding intrusion $H[i]$ from 0 to 1. $H[i]$ is equal to 0 and changing it from 0 to 1 causes no constraint violation.

28.5.5.3 Horizontal Permutation

Horizontal permutation consists in permuting positions x_i^j equal to 1 with positions x_i^j equal to 0 so as to reduce the Manhattan distance associated with H. In this permutation, we keep the same number of intrusion but change their type. The pseudo-code is given by Operator 3.

Operator 3 Horizontal permutation procedure

```

Distinit = Manhattan distance associated with the initial vector H
for each i (i = 1..Na) with H[i]=1 do
  Indicemin=-1 // initialize the variable Indicemin
  // in the loop bellow we look for Indicemin
  for each j (j = 1..Na) with H[j]=0 do
    Distpermutate= Manhattan distance associated with H if we permute H[j] with H[i] ;
    if Distpermutate < Distinit then
      //Update Distinit
      Distinit = Distpermutate
      Indicemin = j;
    end if
    //if we found Indicemin
    if Indicemin != -1 then
      //do the permutation
      H[i] = 0;
      H[j] = -1; // put -1 instead of 1 to avoid repeating the same permutation (at the end of
      the procedure we return -1 to 1)
    end if
  end for
end for

```

28.6 Experimental Results

In order to evaluate the effectiveness of the proposed approach, several tests were carried out using Attack-Events matrices of different sizes. Firstly, we validate our approach using randomly generated instances of different size (28×24 , 100×50 , and 200×100). Subsequently, we use a matrix of real data issued from [12], consisting in a set of 24 attacks scenarios of 28 auditable events. This matrix is used to compare our approach with existing works [12, 5, 3]. The evaluation process requires simulation of a set of attacks. That is, present attacks must be known in advance. In order to evaluate the intrusion detection's quality, we use the following performance measures:

- True positive rate (TPR): proportion of intrusions correctly detected.
- False positive rate (FPR): proportion of intrusions detected but non-existent in the analyzed file.
- Execution time (ET): time used by the approach to find a solution.

Our algorithm attempts to detect intrusions with a minimum FPR (ideally 0%), a maximum TPR (ideally 100%) and minimum execution time (ET). As the proposed approach is stochastic, TPR, FPR and ET are obtained from the average of ten runs. The parameter values α , γ , generation number (G), and population size (P) are experimentally determined. To evaluate the performance measures, various tests were performed and the results are given below.

28.6.1 Performance Validation Using Random Data

To validate the proposed approach, we conducted a series of experiments on randomly generated instances of different sizes. We were considering matrices of dimensions 28×24 , 100×50 and 200×100 .

28.6.1.1 Parameters Setting

To adjust the adaptation parameters α and γ we have performed a parametric study. Figure 28.2a shows the evolution of the intensity A_i according to generation number for different values of α . We have used $\alpha = 0.4, 0.8, 0.9, 0.99$ and we retain the value that gives a moderate decrease of intensity parameter from its initial (maximum close to 1) value towards its final value (minimum close to zero). Figure 28.2b indicates the evolution of the pulse rate r_i , according to generation number for different values of γ . We have used $\gamma = 0.01, 0.1, 0.4, 0.8, 0.9$ and we retain the value that gives a moderate increase of pulse rate parameter from its initial (minimum) value towards its final value r_{max} . Hence, α is set to 0,9 and γ to 0.1.

Thereafter, the performance evaluation was carried out by observing TPR, FPR and TE for different values of the generation number parameter (G) and the population size parameter (P). In these experiments, α is set to 0,9 and γ to 0.1. The number of injected attacks is randomly chosen.

Intrusion Detection Using Matrix AE (28×24)

In Table 28.1, we find that the best value for the population size (P) is 10 (100% TPR and 0% FPR). Table 28.2 shows that good results were obtained for generation number $G = 5$. Thus, we hold the values $P = 10$ and $G = 5$.

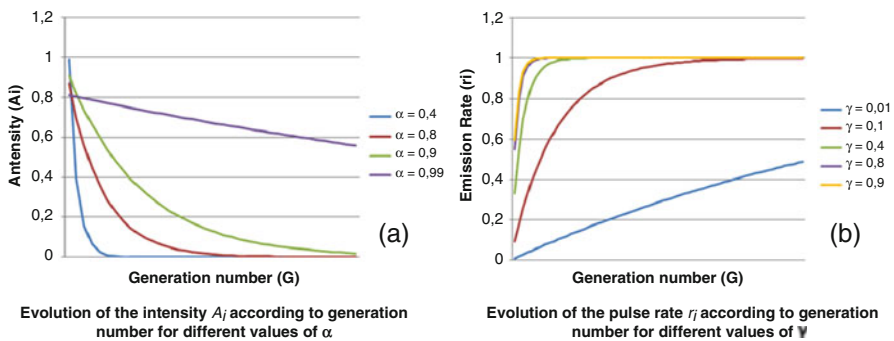


Fig. 28.2 Influence of α and γ on the echolocation parameters

Intrusion Detection Using Matrix AE (100 × 50)

The results given in Tables 28.3 and 28.4 show the influence of the population size (P) and the generation number (G), respectively, on the performance measures (TPR, FPR and ET). In these two tables, we observe that for P=50 and G=20, we get perfect results (100% TPR and 0% FPR).

Table 28.1 Evolution of TPR, FPR and ET according to population size (P) using AE (28 × 24)

Population size (P)	TPR (%)	FPR (%)	ET(s)
6	96.4	5.70	0.00468
10	100	0	0.00624
20	100	0	0.0078

Table 28.2 Evolution of TPR, FPR and ET according to generation number (G) using AE (28 × 24)

Generation number (G)	TPR (%)	FPR (%)	ET(s)
3	97.6	4.28	0.00468
5	100	0	0.00624
20	100	0	0.00624

Table 28.3 Evolution of TPR, FPR and ET according to population size (P) using AE (100 × 50)

Population size (P)	TPR (%)	FPR (%)	ET(s)
5	80.9	16	0.269
10	93	7.5	0.274
50	100	0	0.254
70	100	0	0.404

Table 28.4 Evolution of TPR, FPR and ET according to generation number (G) using AE (100 × 50)

Generation number (G)	TPR (%)	FPR (%)	ET(s)
10	96.3	4	0.840
20	100	0	0.254
100	100	0	0.365

Intrusion Detection Using Matrix AE (200 × 100)

Herein, the experiment is repeated as shown above using larger instances. We test the influence of the population size (P) on the performance measures (TPR, FPR and ET). Table 28.5 shows that for a population size (P) equal to 400, we get a good performance (100% TPR and 0% FPR). The Generation number (G) remained fixed to 20. Thus, we hold the values P = 400 and G = 20.

Table 28.5 Evolution of TPR, FPR and ET according to population size (P) using AE (200 × 100)

Population size (P)	TPR (%)	FPR (%)	ET(s)
200	97.87	7	182.45
400	100	0	41.80
500	100	0	60.22

28.6.1.2 Influence of the Number of Injected Attacks (a) on Intrusion Detection

Table 28.6 shows the influence of the number of injected attacks on the performance measures (TPR, FPR and ET) using the previous 200 × 100 Attack-Events matrix. We observe that the number of injected attacks has no effect on the quality of intrusion detection. However, it influences the running time which grows with the growth of the number of attacks especially for the largest instances.

Table 28.6 Evolution of TPR, FPR and ET according to population size (P) using AE (200 × 100)

Number of attacks (a)	TPR (%)	FPR (%)	ET(s)
2	100	0	0.188
10	100	0	0.482
60	100	0	18.387
80	100	0	41.80

28.6.1.3 The Effect of Manhattan Distance on Intrusion Detection

The effect of the Manhattan distance can be investigated according to Table 28.7. This table shows the performance measures (TPR, FPR and ET) obtained with and without this measure. We compared the results obtained using Binary Bat Algorithm (BBA) [13] without Manhattan distance, with our Enhanced Binary Bat Algorithm (EBBA) that uses Manhattan Distance. We conducted these experiments on Attack-Events matrices of different sizes using the previous parameters. Table 28.7 shows that the proposed Approach (EBBA) improves substantially the intrusion detection quality.

28.6.2 Comparisons of EBBA with BBO, GA and HS Algorithms Using Real Data

In this section, we compare the results obtained by our algorithm (EBBA) with those obtained by GA [12], BBO [5] and HS [3]. This comparison is made using the same real data (matrix AE of dimension 28 × 24) [12]. Table 28.8 indicates the

Table 28.7 Influence of the Manhattan distance on intrusion detection

Matrices/algorithm	BBA (without Manhattan distance)			EBBA (with Manhattan distance)		
	TPR (%)	FPR (%)	ET(s)	TPR (%)	FPR (%)	ET(s)
28 × 24	60	28	0.0109	100	0	0.0078
100 × 60	55	52	7.98	100	0	0.75
200 × 80	66	49	11.63	100	0	6.44
200 × 100	68	53	29.95	100	0	59

performance measures (TPR, FPR and ET) obtained by each algorithm according to the two parameters values (population size and generation number). The rest of EBBA parameters are set to the previous values ($\alpha = 0,9$, $\gamma = 0.1$ and $a = \text{rand} [0, Na]$). We observe that all attacks are detected in all approaches (TPR = 100% and FPR = 0%) except genetic approach. However, our approach needs less time. We observe that the number of generations needed in our method is less than the number of generations needed for the other algorithms because of the intensive exploration and exploitation done during each iteration and because we deal only with feasible solutions.

Table 28.8 Comparisons of EBBA with BBO, GA and HS algorithms

Approach/measures	TPR (%)	FPR (%)	ET(s)	P	G
GA	99.5	0.43	18	500	100
BBO	100	0	0.0705	50	100
HS	100	0	0.0560	30	200
EBBA	100	0	0.00156	10	5

28.7 Conclusion

Security Audit Trail Analysis problem consists in detecting predefined attack scenarios in the audit trails. The proposed work is based on a simplified analysis of the audit trails in which the temporal order of events sequences is not considered. This problem can be formulated as an NP-Hard combinatorial optimization problem [12]. In this paper we propose to use the Bat echolocation approach to solve such problem. We propose an Enhanced Binary Bat Algorithm named EBBA that uses fitness function in conjunction with Manhattan distance to improve the intrusion detection’s quality. The evaluation was performed by observing the performance measures: TPR, FPR and TE using randomly generated instances of different sizes. Experimental results show the effectiveness of the proposed approach to make cor-

rect predictions (100% TPR and 0% FPR). Another experiment was carried out to investigate the effect of the Manhattan distance. The obtained results show that the use of the Manhattan distance improves substantially the intrusion detection's quality. Experiments also show that the number of injected attacks has no effect on the quality of intrusion detection. It influences the running time which grows with the growth of the number of attacks. Comparisons with existing works [12, 5, 3] are made using the same real data (matrix AE of dimension 28×24) [12]. We observe that our approach gives the same good results (100% TPR and 0% FPR) with less time.

References

1. A. Abraham, C. Grosan, Evolving intrusion detection systems, in *Genetic Systems Programming*, ed. by N. Nedjah, L. Mourelle, A. Abraham. Studies in Computational Intelligence, vol. 13 (Springer, Berlin/Heidelberg, 2006)
2. J. Cannady, Artificial neural networks for misuse detection, in *Proceedings of the 98 National Information Systems Security Conference (NISSC'98)* (Virginia Press, Arlington, 1998), pp. 443–456
3. M. Daoudi, Security audit trail analysis using harmony search algorithm, in *Proceeding of the Eighth International Conference on Systems (ICONS)*, Seville, 2013
4. M. Daoudi, M. Ahmed-Nacer, An intrusion detection approach using an adaptative parameter-free algorithm, in *Proceeding of the Ninth International Conference on Systems (ICONS)*, Nice (2014), pp. 178–184
5. M. Daoudi, A. Boukra, M. Ahmed-Nacer, Security audit trail analysis with biogeography based optimization metaheuristic, in *Proceedings of the International Conference on Informatics Engineering & Information Science: ICIES, ICIEIS 2011, Part II, CCIS 252*, ed. by A. Abd Manaf et al. (Springer, Berlin/Heidelberg, 2011), pp. 218–227
6. M. Dass, Lids: a learning intrusion detection system. Master of Science, The University of Georgia, Athens, Georgia, 2003
7. A. Diaz-Gomez, D.F. Hougen, A genetic algorithm approach for doing misuse detection in audit trail files, in *CIC 06 Proceeding of the 15th International Conference on Computing (CIC)* (IEEE Computer Society, Washington, DC, 2006), pp. 329–335
8. Y. Haidong, G. Jianhua, D. Feiqi, Collaborative rfid intrusion detection with an artificial immune system. *J. Intell. Inf. Syst.* **36**(1), 1–26 (2010)
9. C. Koliass, G. Kambourakis, M. Maragoudakis, Swarm intelligence in intrusion detection: A survey. *Comput. Secur.* **30**(8), 625–642 (2011)
10. W. Lee, J. Salvatore, K. Mok, Mining audit data to build intrusion detection models, in *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, New York (1998), pp. 66–72
11. P.G. Majeed, S. Kumar, Genetic algorithms in intrusion detection systems: a survey. *Int. J. Innov. Appl. Stud.* **5**(3), 233–240 (2014)
12. L. Mé, Audit de sécurité par algorithmes génétiques. Ph.D. thesis, Institut de Formation Supérieure en Informatique et de Communication de Rennes, 1994
13. S. Mirjalili, S.M. Mirjalili, X.S. Yang, Binary bat algorithm. *Neural Comput. Appl.* **25**(3), 663–681 (2013)
14. A. Sanjay, R.K. Gupta, Intrusion detection system: a review. *Int. J. Secur. Appl.* **9**(5), 69–76 (2015)

15. E. Tombini, Amélioration du diagnostic en détection d'intrusions: étude et application d'une combinaison de méthodes comportementale et par scénarios. Ph.D. thesis, Institut National des Sciences Appliquées de Rennes, 2006
16. D.P. Vinchurkar, A. Reshamwala, A review of intrusion detection system using neural network and machine learning technique. *Int. J. Eng. Sci. Innov. Technol.* **1**(2), 54–63 (2012)
17. B. Xu, A. Zhang, Application of support vector clustering algorithm to network intrusion detection, in *Proceedings of the International Conference on Neural Networks and Brain ICNN & B'05*, Beijing, vol. 2 (2005), pp. 1036–1040
18. X.S. Yang, A new metaheuristic bat-inspired algorithm, in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, ed. by J.R. Gonzalez, et al. vol. 284 (Springer, Berlin/Heidelberg, 2010), pp. 65–74
19. X.S. Yang, Bat algorithm for multi-objective optimisation. *Int. J. Bio-Inspired Comput.* **3**(5), 267–274 (2011)

Index

- Air traffic, 219
- Aircraft, 219
- Airline routes, 369
- Analytic Network Process, 179
- Attack scenarios, 477
- Attacks risks, 477

- Bat Algorithm, 477
- Bi-objective Optimisation, 256
- Big data, 447
- Bin Packing Problem, 333
- Binary metaheuristic, 447

- Car sequencing, 59
- Casablanca, 369
- Cell formation problem, 151
- Classification error, 447
- Column generation, 333
- Communication delays, 385
- Completion time, 385
- Connectivity, 163
- Coverage, 163, 463
- Covering tour problem, 255
- Cuckoo Search, 429
- Cuckoo search algorithm, 151

- DEMATEL, 179
- Deployment, 463
- Disaster management, 254
- Domain barrier, 75
- Dynamic TRSP, 347

- Evacuation problem, 286

- Facility location, 255
- Feature, 429
- Feature selection, 447
- Flight schedule, 369
- French airspace, 219
- Fuzzy logic, 415
- Fuzzy makespan, 415

- Genetic Algorithm, 163
- Genetic algorithm, 125
- GRASP, 305
- Gravitational search algorithm, 447
- Greedy heuristics, 333
- Greedy Randomized Adaptive Search algorithm, 305
- Group Technology, 151

- Hidden Markov Model, 1
- Hill-climbing, 109
- Humanitarian logistics, 254
- Hyper-heuristics, 75

- Identification, 429
- Interconnected metaheuristic, 91
- Intrusion detection, 477
- Inventory management, 59

- Job shop scheduling, 415

- Large Neighborhood Search, 347
- Layer, 75
- Local search, 43

- Makespan, 385, 415
- Manhattan distance, 477
- Match-making problem, 145
- Matching, 429
- Monte Carlo, 163
- Moving, 429
- Multi-core, 91
- Multi-level strategy, 463
- Multi-objective, 17
- Multi-objective problem, 369
- Multi-Objective PSO, 199
- Multiobjective, 415
- Multiprocessor, 385

- Neighborhood, 109
- Neighborhoods, 43
- Network flow, 285
- Network reliability, 163
- NSGA-II, 255, 463

- OscaR.cbcls solver, 43

- Parallel architecture, 91
- Parallel-Oriented Solver Language, 91
- Pareto Frontier, 199
- Particle Swarm Optimization, 199
- Particle swarm optimization, 1, 385, 447
- Person, 429
- Pivoting rules, 109
- Possibilistic framework, 18
- PSO, 1, 385

- Radio frequency identification, 463
- Reoptimization, 399
- Resource allocation, 59
- RFID, 463
- RFID reader problem, 463
- Robustness, 415
- Royal Air Maroc, 369
- Runtime analysis, 126

- Scheduling, 59, 385
- Security Audit Trail, 477
- Selection strategies, 109
- Sensor Networks, 163
- Shift scheduling problem, 399
- Shortest path problem, 125
- Simulated Annealing, 163
- Skills, 347
- SMPSO Algorithm, 369
- Spare parts, 347
- Strip Algorithm, 319
- Supplier selection, 179
- Supply chain, 179
- Surveillance Patrol, 305

- Tabu search, 59
- Technician Routing and Scheduling Problem, 347
- Time window, 333
- Time windows, 347
- TOPSIS, 179
- Traceability, 463
- Tracking, 429
- Traffic Assignment Problem, 199
- Traffic network design, 199
- Traffic Simulation, 199
- Trajectory, 219
- Trajectory-Based Operations, 219
- Travelling Salesman Problem, 319

- Uncertain processing times, 415
- Uncertainty, 18

- Vehicle Routing Problem, 305
- Vehicle routing problem, 18, 255
- Video, 429
- VRP, 305

- Wireless Sensor, 163