Journal Subline

LNCS 10780

Jaap van den Herik · Ana Paula Rocha
Joaquim Filipe
Guest Editors

Transactions on
Computational Collective Intelligence XXVIII

Ngoc Thanh Nguyen • Ryszard Kowalczyk
Editors-in-Chief

Springer

Lecture Notes in Computer Science 10780

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

Editorial Board

David Hutchison
Lancaster University, Lancaster, UK
Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA
Josef Kittler
University of Surrey, Guildford, UK
Jon M. Kleinberg
Cornell University, Ithaca, NY, USA
Friedemann Mattern
ETH Zurich, Zurich, Switzerland
John C. Mitchell
Stanford University, Stanford, CA, USA
Moni Naor
Weizmann Institute of Science, Rehovot, Israel
C. Pandu Rangan
Indian Institute of Technology Madras, Chennai, India
Bernhard Steffen
TU Dortmund University, Dortmund, Germany
Demetri Terzopoulos
University of California, Los Angeles, CA, USA
Doug Tygar
University of California, Berkeley, CA, USA
Gerhard Weikum
Max Planck Institute for Informatics, Saarbrücken, Germany

More information about this series at http://www.springer.com/series/8851

Ngoc Thanh Nguyen · Ryszard Kowalczyk
Jaap van den Herik · Ana Paula Rocha
Joaquim Filipe (Eds.)

Transactions on Computational Collective Intelligence XXVIII

Springer

Editors-in-Chief

Ngoc Thanh Nguyen
Wrocław University of Technology
Wrocław
Poland

Ryszard Kowalczyk
Swinburne University of Technology
Hawthorn, VIC
Australia

Guest Editors

Jaap van den Herik
Leiden University
Leiden
The Netherlands

Ana Paula Rocha
Universidade do Porto
Porto
Portugal

Joaquim Filipe
INSTICC
Polytechnic Institute of Setúbal
Setubal
Portugal

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISSN 2190-9288 ISSN 2511-6053 (electronic)
Transactions on Computational Collective Intelligence
ISBN 978-3-319-78300-0 ISBN 978-3-319-78301-7 (eBook)
https://doi.org/10.1007/978-3-319-78301-7

Library of Congress Control Number: 2018940914

© Springer International Publishing AG, part of Springer Nature 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 the registered company Springer International Publishing AG
part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The present special issue of the journal of *Transactions on Computational Collective Intelligence* (TCCI) includes extended and revised versions of a set of 11 selected papers from the International Conference on Agents and Artificial Intelligence, ICAART 2016 and 2017 editions.

The multidisciplinary areas of agents and artificial intelligence involve a large number of richly variegated researchers. They devote themselves to the study of theoretical and practical issues related to areas such as multi-agent systems, software platforms, agile management, distributed problem solving, distributed AI in general, knowledge representation, planning, learning, scheduling, perception, data mining, data science, reactive AI systems, evolutionary computing, and other topics related to intelligent systems and computational intelligence.

This special issue comprises 11 research papers with novel concepts and applications, which were selected after a strict reviewing process. Our main criterion was interesting and relevant topics of current research on agents and artificial intelligence.

The first six papers are revised and extended versions of papers presented at ICAART 2016. The issue starts with two papers related to user preference modelling, the first one is from an individual point of view and the second one is from a group recommendation perspective. The first paper is entitled "A New Approach for Learning User Preferences for a Ridesharing Application," authored by Mojtaba Montazery and Nic Wilson, and focuses on the learning aspect of user preferences. The second paper is entitled "An Altruistic-Based Utility Function for Group Recommendation," by Silvia Rossi, Francesco Cervone, and Francesco Barile, and focuses on models of other-regarding preferences. Here it is remarked that altruism, fairness, and reciprocity strongly motivate many people. Then two papers deal with multi-agent cooperation, either in a game context or in a more general planning context. The first entitled "Two-Stage Reinforcement Learning Algorithm for Quick Cooperation in Repeated Games," by Wataru Fujita et al., highlights the importance of "trust" in reinforcement learning. The other paper, entitled "Recursive Reductions of Action Dependencies for Coordination-based Multiagent Planning," by Jan Tožička, Jan Jakubův, and Antonín Komenda, describes how to extract internally dependent actions during multiagent planning and how to make the planning process more efficient by taking advantage of the computed dependencies. In paper five and six, soft-computing methodologies are described and applied, namely, in an applied-research paper entitled "Controlling a Single Transport Robot in a Flexible Job Shop Environment by Hybrid Metaheuristics," by Houssem Nouri et al., and in a more theoretical paper, focusing on continuous optimization tasks, entitled "Can Evolution Strategies Benefit from Shrinkage Estimators?" by Silja Meyer-Nieberg and Erik Kropat.

The other five papers are revised and extended versions of papers presented at ICAART 2017. Two papers are related to the topic of agent cooperation, two papers to the topic of adaptation and one paper to allocation strategies. The first cooperation paper is authored by Jean-Claude Heudin, entitled “An Emotional Multi-Personality Architecture for Intelligent Conversational Agents.” The second cooperation paper is authored by João Marinheiro and Henrique Cardoso, entitled “Towards General Cooperative Game Playing.” The first adaptation paper is entitled “Comparing the Effects of Disturbances in Self-Adaptive Systems,” by Sven Tomforde et al. The second adaptation paper is entitled “Analysis of Perceived Helpfulness in Adaptive Autonomous Agent Populations” authored by Mirgita Frasheri, Baran Çürüklü, and Mikael Ekström. Finally, the strategy paper is entitled “Evaluating Task-Allocation Strategies for Emergency Repair in MAS,” authored by Hisashi Hayashi. This paper addresses the robustness of multi-agent system types in a comparative analysis. It shows how task-allocation algorithms in emergency situations are important for avoiding the failure of the whole system.

We believe that all papers published in this special issue will serve as a guiding reference for students, researchers, engineers, and practitioners who perform research in the areas of agents and artificial intelligence. Moreover, we hope that the readers will find new inspiration for their research and may join the ICAART community in the future.

Finally, we would like to thank all authors for their contributions and all reviewers for supporting us in ensuring the quality of this publication. Of course, we would like to express our gratitude to the LNCS editorial staff of Springer, in particular to Prof. Ryszard Kowalczyk for his patience and availability during this process.

April 2018

Jaap van den Herik
Ana Paula Rocha
Joaquim Filipe

Transactions on Computational Collective Intelligence

This Springer journal focuses on research in applications of the computer-based methods of computational collective intelligence (CCI) and their applications in a wide range of fields such as the Semantic Web, social networks and multi-agent systems. It aims to provide a forum for the presentation of scientific research and technological achievements accomplished by the international community.

The topics addressed by this journal include all solutions to real-life problems for which it is necessary to use CCI technologies to achieve effective results. The emphasis of the papers published is on novel and original research and technological advancements. Special features on specific topics are welcome.

Editor-in-Chief

Ngoc Thanh Nguyen	Wroclaw University of Technology, Poland

Co-Editor-in-Chief

Ryszard Kowalczyk	Swinburne University of Technology, Australia

Editorial Board

John Breslin	National University of Ireland, Galway, Ireland
Longbing Cao	University of Technology Sydney, Australia
Shi-Kuo Chang	University of Pittsburgh, USA
Oscar Cordon	European Centre for Soft Computing, Spain
Tzung-Pei Hong	National University of Kaohsiung, Taiwan
Gordan Jezic	University of Zagreb, Croatia
Piotr Jędrzejowicz	Gdynia Maritime University, Poland
Kang-Huyn Jo	University of Ulsan, Korea
Yiannis Kompatsiaris	Centre for Research and Technology Hellas, Greece
Jozef Korbicz	University of Zielona Gora, Poland
Hoai An Le Thi	Lorraine University, France
Pierre Lévy	University of Ottawa, Canada
Tokuro Matsuo	Yamagata University, Japan
Kazumi Nakamatsu	University of Hyogo, Japan
Toyoaki Nishida	Kyoto University, Japan
Manuel Núñez	Universidad Complutense de Madrid, Spain
Julian Padget	University of Bath, UK
Witold Pedrycz	University of Alberta, Canada
Debbie Richards	Macquarie University, Australia
Roman Słowiński	Poznan University of Technology, Poland

Edward Szczerbicki	University of Newcastle, Australia
Tadeusz Szuba	AGH University of Science and Technology, Poland
Kristinn R. Thorisson	Reykjavik University, Iceland
Gloria Phillips-Wren	Loyola University Maryland, USA
Sławomir Zadrożny	Institute of Research Systems, PAS, Poland
Bernadetta Maleszka	Assistant Editor, Wroclaw University of Technology, Poland

Contents

A New Approach for Learning User Preferences for a Ridesharing Application 1
Mojtaba Montazery and Nic Wilson

An Altruistic-Based Utility Function for Group Recommendation 25
Silvia Rossi, Francesco Cervone, and Francesco Barile

Two-Stage Reinforcement Learning Algorithm for Quick Cooperation in Repeated Games 48
Wataru Fujita, Koichi Moriyama, Ken-ichi Fukui, and Masayuki Numao

Recursive Reductions of Action Dependencies for Coordination-Based Multiagent Planning 66
Jan Tožička, Jan Jakubův, and Antonín Komenda

Controlling a Single Transport Robot in a Flexible Job Shop Environment by Hybrid Metaheuristics 93
Houssem Eddine Nouri, Olfa Belkahla Driss, and Khaled Ghédira

Can Evolution Strategies Benefit from Shrinkage Estimators? 116
Silja Meyer-Nieberg and Erik Kropat

An Emotional Multi-personality Architecture for Intelligent Conversational Agents 143
Jean-Claude Heudin

Towards General Cooperative Game Playing 164
João Marinheiro and Henrique Lopes Cardoso

Comparing the Effects of Disturbances in Self-adaptive Systems - A Generalised Approach for the Quantification of Robustness 193
Sven Tomforde, Jan Kantert, Christian Müller-Schloer, Sebastian Bödelt, and Bernhard Sick

Analysis of Perceived Helpfulness in Adaptive Autonomous Agent Populations 221
Mirgita Frasheri, Baran Çürüklü, and Mikael Ekström

Evaluating Task-Allocation Strategies for Emergency Repair in MAS 253
Hisashi Hayashi

Author Index 275

A New Approach for Learning User Preferences for a Ridesharing Application

Mojtaba Montazery and Nic Wilson

School of Computer Science and IT, Insight Centre for Data Analytics,
University College Cork, Cork, Ireland
`{mojtaba.montazery,nic.wilson}@insight-centre.org`

Abstract. Ridesharing has the potential to relieve some transportational issues such as traffic congestion, pollution and high travel costs. In this paper, we focus on the process of matching drivers and prospective riders more effectively, which is a crucial challenge in ridesharing. A novel approach is proposed in ride-matching which involves learning user preferences regarding the desirability of a choice of matching; this could then maintain high user satisfaction, thus encouraging repeat usage of the system. An SVM inspired method is developed which is able to learn a scoring function from a set of pairwise comparisons, and predicts the satisfaction degree of the user with respect to specific matches. To assess the proposed approach, we conducted some experiments on a commercial ridesharing data set. We compare the proposed approach with five rival strategies and methods, and the results clearly show the merits of our approach.

Keywords: User preference learning · Dynamic ridesharing

1 Introduction

Ridesharing (also *carpooling* and *lift-sharing*) is a mode of transportation in which individual travellers share a vehicle for a trip. Increasing the number of travellers per vehicle trip, by effective usage of spare car seats, may of course enhance the efficiency of private transportation, and contribute to reducing traffic congestion, fuel consumption, and pollution. Moreover, ridesharing allows users to split travel costs such as fuel, toll, and parking fees with other individuals who have similar itineraries and time schedules. Conceptually, ridesharing is a system that can combine the flexibility and speed of private cars with the reduced cost of fixed-line systems such as buses or subways (Furuhata et al. 2013, Agatz et al. 2011).

Ridesharing is quite an old concept; it was first used in the USA during World War II to conserve resources for the war. It reappeared as a result of the oil crisis in 1970s which led to the emergence of the first ridesharing algorithms. Nevertheless, ridesharing usage declined drastically between the 1970s and the 2000s due to the decrease in the price of fuel and vehicle ownership cost (Chan and Shaheen 2012).

© Springer International Publishing AG, part of Springer Nature 2018
N. T. Nguyen et al. (Eds.): TCCI XXVIII, LNCS 10780, pp. 1–24, 2018.
https://doi.org/10.1007/978-3-319-78301-7_1

Furthermore, there are some challenges that have inhibited wide adoption of ridesharing. A few of the most important of those are listed as follows:

Riding with Strangers. Surveys suggest that there is little interest in sharing a ride with strangers, because of personal safety concerns. This phenomenon is called *Stranger Danger* and could be alleviated by incorporation of social networks (Amey et al. 2011, Furuhata et al. 2013). (Chaube et al. 2010) conducted a survey among students of a university which shows that while only 7% of participants would accept rides from a stranger, 98% and 69% would accept rides from a friend and the friend of a friend, respectively.

Reliability of Service. One of the largest behavioural challenges is the perception of low reliability in ridesharing arrangements; the parties may not necessarily follow through on the agreed-upon ride. For instance, if the driver has an unexpected appointment or emergency, the passenger may be left with no ridesharing option; or, from the other side, drivers might be required to wait because of a passenger being late (Amey et al. 2011).

Schedule Flexibility. The lack of schedule flexibility has been one of the longest running challenges in ridesharing arrangements. Drivers and passengers often agree to relatively fixed schedules and meeting locations, not allowing much flexibility. It is interesting to note that increased flexibility and increased reliability in ridesharing arrangements are often conflicting objectives (Amey et al. 2011).

Ride Matching. Optimally matching riders and drivers—or at least getting a good match—is among the most important challenges to be overcome. This can lead to a complicated optimisation problem due to the large number of factors involved in the objective function. We will discuss this aspect of ridesharing further in Sect. 2.

Despite the above barriers to ridesharing, the demand for ridesharing services has increased again sharply in recent years, generating much interest along with media coverage (Saranow 2002). This boost in ridesharing is mainly associated with a relatively new concept in ridesharing, *dynamic* or *real-time* ridesharing. Dynamic ridesharing refers to a system which supports an automatic ride-matching process between participants at short notice or even en-route (Agatz et al. 2012).

Technological advances, both hardware and software, are key enablers for dynamic ridesharing. The first influential fact is that the smartphones are becoming increasingly popular (Emarketer 2014, Smith 2015). The first impact of smartphones on ridesharing is that they provide an infrastructure on which a ridesharing application can run, replacing the old-fashioned, sometimes not so convenient, approaches such as phone or website. More importantly, smartphones are usually equipped with helpful communication capabilities, including Global Positioning System (GPS) (Zickuhr 2012) and network connectivity (Duggan and Smith 2013).

Dynamic ridesharing by its nature is able to ease some aspects of existing challenges in traditional ridesharing. For example, tracking participants by means of GPS could mitigate safety concerns or increase the reliability. In terms of flexibility, since dynamic ridesharing does not necessarily require long-term commitment, users have the option to request a trip sharing day-by-day or whenever they are pretty sure about their itinerary.

Even though the above advancement in technology could be beneficially available, ridesharing is still in short supply. In this study, we focus on the ride-matching problem which is central to the concept. However, we are mindful of the fact that there are a number of other challenges that should be dealt with to accomplish the ultimate success of ridesharing. The next example explains a simple scenario of matching between two individuals.

Example 1. Suppose Alice is going to drive from A to B, and her driving speed is around $80^{km}/h$ on average. Another individual, Bob, needs to travel from C to D. To do this, he goes up from C to E with a taxi to get a train; then, he gets off the train at F, and finally he takes a bus to reach to his destination D. His trip will take approximately 115 min. The schema of routes and their distances are depicted in Fig. 1.

If a ride-matching between them is suggested, Alice needs to drive to C to pick up Bob, and then drops him off at D, and proceeds to her destination B. By doing this, the total system-wide vehicle-mileage will be $(16 + 112 + 12 =)$ 140 km, and the total system-wide travel time will be 189 min because Alice's trip will take $(140 \times {}^{60}/_{80} =)$ 105 min and Bob's travel will last $(112 \times {}^{60}/_{80} =)$ 84 min.

On the other hand, if they travel individually, the total system-wide vehicle-mileage (including bus, taxi and train) will be 238 km since Bob travels $(10 + 100 + 8 =)$ 118 km and Alice 120 km, and the total system-wide travel time will be $(120 \times {}^{60}/_{80} + 115 =)$ 205 min.

Thus, if this matching takes place, $(238 - 140 =)$ 98 km travel distance will be saved, and $(205 - 189 =)$ 16 min travel time will be saved. In this scenario, Alice will drive 20 km (15 min) more than her original route (AB), which is usually compensated with a fair payment by Bob. □

The rest of the paper is structured as follows. In Sect. 2, we model the automated ride-matching problem and explain how user preferences could be considered in the matching process. In Sect. 3, a method to learn user preferences is described in detail. Section 4 evaluates the presented approach on a real

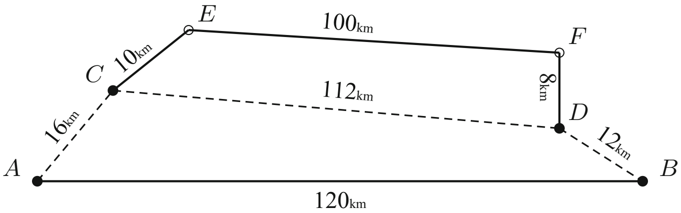


Fig. 1. The routes of Bob and Alice's trips, described in Example 1.

ridesharing database. Finally, in Sect. 5, we summarize the main remarks and discuss some directions for future research.

2 Automated Ride-Matching

The essential element of dynamic ridesharing is the automation of the ride-matching process, which allows trips to be arranged at short notice with minimal effort from participants. This means that a system helps riders and drivers to find suitable matches and facilitates the communication between participants (Hwang et al. 2006, Agatz et al. 2012).

In order to model the matching problem, two disjoint types of ridesharing request are considered: a set of requests in which the owner of the request are drivers (D), and requests created by riders (R). Hence, all trip requests could be represented by the set $\mathcal{S} = R \cup D$. Then, ridesharing requests are represented as a bipartite graph $G = (D, R, E)$, with E denoting the edges of the graph. This setting is extendable for the case when some participants are happy with being either a driver or a rider.

This graph becomes a weighted graph by assigning a weight c_{ij} to the edge (D_i, R_j), where $D_i, R_j \in \mathcal{S}$. Generally speaking, c_{ij} quantifies how much is gained by matching D_i and R_j. This weight is usually a composition of overall system's benefits. For representing the system's benefits—which ultimately could result in less pollution, traffic congestion etc.,—two measures are often mentioned in the related studies; the saved travel distance (d_{ij}) and the saved travel time (t_{ij}) which are obtained from the match (D_i, R_j) (Calvo et al. 2004, Winter and Nittel 2006). For instance, c_{ij} could be defined to be $d_{ij} + t_{ij}$ (or some other linear combination). To represent infeasibility of matching between D_i and R_j, c_{ij} could be assigned to be a very small number (e.g., $-\infty$). Figure 2 depicts a sample matching graph where c_{ij} only considers the saved travel distance measure.

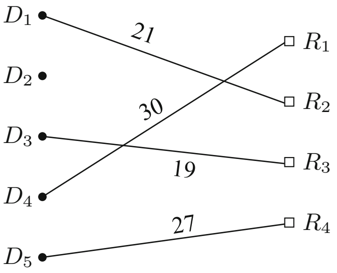
For finding optimal matchings, one approach popular in the literature is solving an optimisation problem in which the sum of the benefits from the proposed matchings is maximised (Agatz et al. 2011). To do this, a binary decision variable x_{ij} is introduced that would be 1 when the match (D_i, R_j) is proposed, and 0 if not. Then, the objective function to be maximised is $\sum_{i,j} x_{ij} c_{ij}$. After running the solver, a fixed schedule is proposed to users as the optimal solution.

However, this approach neglects a crucial requirement of a practical system, that is, getting users' confirmation before fixing a ride-share for them. Although earlier we emphasised the concept of automation in ride-matching which attempts to minimise users' efforts, we believe that it couldn't be fully automatic. In fact, it is hard in practice to convince users to share their ride with somebody without their final agreement. For example in Fig. 2, it would be illogical to send a message to D_1 saying: "According to our computation, the best match for you is R_2. So, you are supposed to give him a ride tomorrow morning at 8:30."

For this reason, in this study, we suggest a novel attitude towards ride-matching problems, by looking at the problem as a recommendation system

	R₁	**R₂**	**R₃**	**R₄**
D_1	22	㉑	-	28
D_2	14	-	10	-
D_3	12	10	⑲	8
D_4	㉚	-	18	-
D_5	-	7	25	㉗

(a)



(b)

Fig. 2. (a) Each cell indicates the amount of saved travel distance (km) for that match, where infeasible matchings are shown with hyphen (-). (b) Having assumed that $c_{ij} = t_{ij}$, the optimal matching is drawn. The total saved travel distance for this whole matching is 97 km, which is the maximum.

rather than an optimisation problem. In this setting, the system just recommends a set of best possible matchings to each individual with respect to the weights c_{ij}.

As previously mentioned, c_{ij} only considers the system's benefits, and to express users' limitations, the common strategy is posing some constraints on the matching graph (Herbawi and Weber 2012, Agatz et al. 2011). Nevertheless, there are two main shortcomings that have not been fully addressed by adoption of this strategy:

Only Hard Constraints. The fact is that users might sacrifice some of their desires in favour of other ones. For instance, a rider who likes smoking in the car may decide to share his trip in a non-smoking vehicle due to other favourable conditions. Therefore, having soft constraints instead of hard ones may often be more plausible. Posing soft constraints could be rephrased as considering users' preferences, the term that is mainly used in this paper.

Eliciting Complicated Parameters. In order to set constraints, there is a prerequisite to ask users to specify several parameters explicitly, such as earliest possible departure time, latest possible arrival time and maximum excess travel time (Baldacci et al. 2004, Agatz et al. 2011, Amey et al. 2011). However, elicitation of such parameters for every trip is quite impractical in real world situations. As an example, users can simply state their departure time, but finding how flexible they are with that time is not a straightforward task. Moreover, some participants may be hesitant or unwilling to disclose certain preferences for privacy reasons.

Our solution for attempting to overcome the aforementioned issues is redefining c_{ij} in such a way that not only does c_{ij} incorporate the overall system's benefits, but also takes into account participants' desires, making the hard constraints unnecessary. Actually, c_{ij} should also represent how happy two individuals (D_i and R_j) are with being matched with each other. While positional and temporal elements of a ride-share usually play a more important role in forming users' satisfaction degree, users' social preferences such as the other party's reputation and gender, smoking habit and so forth are also suggested as relevant components (Ghoseiri et al. 2011). To mathematically illustrate, assume $c_{ij} = (w_{ij}^{(D)} \cdot w_{ij}^{(R)}) \cdot (d_{ij} + t_{ij})$ where $w_{ij}^{(D)}$ indicates how favourable sharing the ride with the rider j for the driver i is, and the converse for $w_{ij}^{(R)}$.

In order to find $w_{ij}^{(D)}$ (resp. $w_{ij}^{(R)}$), we suggest learning the preferences of the user i (resp. j) from his/her past ridesharing records. For instance, we can learn from the previous choices of a particular user that he is normally more flexible with time than location. We will discuss more about this subject in Sect. 2.1. As a result of learning weights, as well as modeling soft constraints in the form of users' preferences, there is no need to elicit those cumbersome parameters from participants anymore.

2.1 Learning Weights

In this section, the mechanism of learning users' preferences from the previous ridesharing records will be characterised. To give an intuitive sense, we start with an example.

Example 2. Take Alice from Example 1 who has a daily driving trip from A to B (e.g., every day at 8:30). This trip request is denoted by D_1.

On Day 1, the system recommends three matching opportunities, namely R_1, R_2 and R_3. Alice declines R_1, accepts R_2 and leaves R_3 without any response. Ignoring a recommended match may potentially mean that the user had been tentative about it.

On the second day, two recommendations, R_4 and R_5, are suggested to her. R_4 is declined and R_5 is initially accepted. While Alice was waiting for the response of the other party in R_5, she receives another suggestion, R_6; she then changes her mind about R_5 and cancels that one, and instead accepts R_6.

On the third day, the system has found two feasible matches which are R_7 and R_8. The goal is to evaluate how desirable these two opportunities are for Alice regarding her trip D_1. Using the notation of the previous section, we would like to find $w_{1,7}^{(D)}$ and $w_{1,8}^{(D)}$ which are finally incorporated in the calculation of $c_{1,7}$ and $c_{1,8}$, respectively. Table 1 summarizes this example. □

Table 1. The scenario described in Example 2 is summarized here. Alice's responses are Accepted(A), Cancelled(C), Ignored(I) and Declined(D).

Day 1	Day 2	Day 3 (Today)
(R_1, D)	(R_4, D)	$R_7(w_{1,7}^{(D)} =?)$
(R_2, A)	(R_5, C)	$R_8(w_{1,8}^{(D)} =?)$
(R_3, I)	(R_6, A)	

At first glance, the problem might seem to be a classification problem because the supervision (labels) is in the form of classes (i.e., Accepted, Declined etc.). However, a closer look suggests that learning a classifier may well not be a good choice. The first reason is that there is a logical order between classes in the current case (e.g., Accepted has a highest value and Declined the lowest value), whereas classification is often applied when there is no ordering between class labels. Secondly, a classifier will predict a class label whereas here, a scalar value is required.

If each class label is replaced by an appropriate number which keeps the natural ordering, the problem could be considered as a regression problem. For instance, the number 1 might be assigned for Accepted, 0.67 for Cancelled, 0.33 for Ignored and 0 for Declined. In spite of the fact that learning a regressor has none of those defects mentioned about classification, we believe that it still suffers from the following flaws:

- The user's response to a recommended opportunity not only depends on the properties of that particular case, but also depends on other rival opportunities. Taking Example 2 to illustrate, it is not necessarily the case that if R_2 existed in the suggestion list on day 2, it would certainly be accepted again, because Alice accepted R_2 in presence of R_1 and R_3 which does not guarantee its acceptance when R_5 and R_6 are available.
- Two class labels do not necessarily have the same desirability distance for all instances. Consider Example 2; assigning value numbers to classes as described above suggests that the difference between R_1 and R_2 from the first day, and R_4 and R_6 from the second day are both 1. However, the only thing that is known is that Alice preferred R_2 to R_1 and R_6 to R_4, not the extent of the difference.

To address the above issues, we suggest considering the supervision in the form of qualitative preferences. This means that instead of assigning value numbers, sets of *pairwise comparisons* among alternatives are derived from the user's choices. The following set of preferences could be formed for Example 2:

$$\lambda = \{(R_2 \succ R_3), (R_3 \succ R_1), (R_2 \succ R_1), (R_6 \succ R_5), (R_5 \succ R_4)\}, (R_6 \succ R_4).$$

In the next section, a model will be proposed to learn a scoring function by making use of this kind of preferences set.

3 Learning Model: SVPL

3.1 Basic Formulation

As described above, the primary capability of the method being developed in this section should be learning a scoring function from a set of pairwise comparisons, expressed between several alternatives. This scoring function can generate a value number for each alternative (i.e., ridesharing trip opportunity), measuring the expected desirability degree of that alternative for the user.

Mathematically speaking, the set of alternatives is expressed as $\mathcal{X}$ which is a finite subset of $\mathbb{R}^d$, where d is the dimension of the features vector (with each feature representing a different property of the trip). We formulate the pairwise comparisons set as $\lambda = \{(X_i \succ Y_i) | i \in \{1, \ldots, n\},\ X_i, Y_i \in \mathcal{X}\}$. Then, the goal is finding a function $f(X) : \mathbb{R}^d \rightarrow \mathbb{R}$ which maps a features vector to a scalar value and is in agreement with the preferences set λ, i.e., if $X_i \succ Y_i$ is in λ then we aim to have $f(X_i) > f(Y_i)$. In order to achieve this goal, we developed a derivation of conventional SVM which eventually turns out to be similar to the SVMRank approach proposed in (Joachims 2002).

In the rest of this section, the notation λ_i (corresponding to $(X_i \succ Y_i)$) represents the *preference point* in $\mathbb{R}^d$, given by $\lambda_i = X_i - Y_i$. We say that preferences points are consistent if there is a scoring function f satisfies the following condition:

$$\forall i = 1, \ldots, n, \quad f(X_i) > f(Y_i). \tag{1}$$

In the current section, we assume that all preferences points are consistent, meaning there is at least one scoring function able to satisfy Eq. (1). In Sect. 3.2, we consider the inconsistent case.

Another assumption being made in this section is that the scoring function is a linear function, and thus, $f(\lambda_i) > 0$ (since $f(X_i) - f(Y_i) > 0$). We will consider the case of a non-linear scoring function in Sect. 3.3.

With respect to these two assumptions, if $h = \{X : f(X) = 0\}$, then the hyperplane h (which is a line in the 2-dimensional space) should clearly include the origin, and all preferences points (i.e., λ_i) should be in the associated positive open half-space, given by h. This condition can be mathematically written as:

$$\omega \cdot \lambda_i > 0 \ \ \forall i, \tag{2}$$

where ω is the normal vector to the hyperplane h. Two examples of plausible hyperplanes (h_{ω_1} and h_{ω_2}) for a consistent set of preferences points are illustrated in Fig. 3.

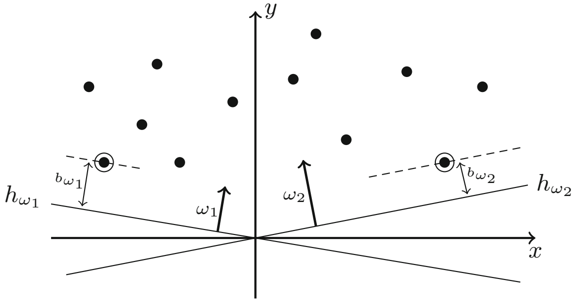


Fig. 3. Two samples of plausible hyperplanes (h_{ω_1} and h_{ω_2}) and their associated normal vectors (ω_1 and ω_2) for a set of consistent preferences points are illustrated. b_{ω_1} and b_{ω_2} are the margins of h_{ω_1} and h_{ω_2}, respectively.

The *margin* of a hyperplane, denoted by b, is defined as the distance from the hyperplane to the closest preference point; for the hyperplane h_ω, it is equal to:

$$b_\omega = \frac{\min_i \omega \cdot \lambda_i}{\|\omega\|}, \tag{3}$$

where $\|\omega\|$ is the Euclidean norm of ω ($\|\omega\|^2 = \omega \cdot \omega$).

Based on the principal idea in SVM (Burges 1998), among all hyperplanes that can meet the condition stated in Eq. (2), we look for the hyperplane that produces the largest margin, since it seems reasonable that a greater marginal distance from the condition boundary is more desirable. Thus, in order to find the optimal ω (say ω^*) and consequently $f(X)$, we need to solve an optimisation problem as follows:

$$\omega^* = \arg\max_\omega \frac{\min_i \omega \cdot \lambda_i}{\|\omega\|} \tag{4a}$$

subject to

$$\omega \cdot \lambda_i > 0 \quad \forall i. \tag{4b}$$

Let $a_\omega = \min_i \omega \cdot \lambda_i$, where according to Eq. (4b) a_ω should be a strictly positive number. Then we have:

$$\omega^* = \arg\max_\omega \frac{a_\omega}{\|\omega\|} \tag{5a}$$

subject to

$$\omega \cdot \lambda_i \geq a_\omega > 0 \qquad \forall i. \tag{5b}$$

Let $\omega' = \frac{\omega}{a_\omega}$, where clearly ω', like ω, is a normal vector to the hyperplane h_ω. As a result, the problem can be simplified by replacement of ω with the term $a_\omega \omega'$. That means the objective function (Eq. 5a) will be:

$$\frac{a_\omega}{\|\omega\|} = \frac{a_\omega}{a_\omega \|\omega'\|} = \frac{1}{\|\omega'\|}$$

and the constraint (Eq. 5b) is modified like this:

$$\omega' \cdot \lambda_i \geq 1$$

Hence, the problem is reformulated as the following:

$$\omega'^* = \arg\max_{\omega'} \frac{1}{\|\omega'\|} \tag{6a}$$

subject to

$$\omega' \cdot \lambda_i \geq 1 \quad \forall i. \tag{6b}$$

Note that ω'^* found from Eq. (6) is not necessarily equal to ω^* found from Eq. (5), but as described above, both give an equivalent hyperplane ($h_\omega \equiv h_{\omega'}$). So, without loss of generality, we can use the symbol ω instead of ω' in Eq. (6).

To have a more standard form, the arrangement of the problem is reformulated as a minimisation problem in this manner:

$$\omega^* = \arg\max \frac{1}{2}\|\omega\|^2 \tag{7a}$$

subject to

$$1 - \omega \cdot \lambda_i \geq 0 \quad \forall i. \tag{7b}$$

Now, this is a convex quadratic programming problem, because the objective function is itself convex, and those points which satisfy the constraints are also from a convex set (a set of linear constraints define a convex set).

In Sect. 3.3, more explanation will be given as to why we are interested in the form of problem formulation (both objective function and constraints) in which preferences points only appear in the form of pairwise dot products. To fulfill this demand, we switch to the *Lagrangian Dual Problem.*

For our problem, the *Lagrangian function* which is basically obtained by augmenting the objective function with a weighted sum of the constraint functions, is:

$$\mathcal{L}(\omega, \mu) = \frac{1}{2}\|\omega\|^2 + \sum_{i=1}^{n} \mu_i \left(1 - \omega \cdot \lambda_i\right) \tag{8}$$

where μ_i is referred as the *Lagrange multiplier* associated with the ith inequality constraint $1 - \omega \cdot \lambda_i \leq 0$ (Boyd and Vandenberghe, 2004 Chap. 5).

The optimisation problem stated in (Eq. 7) is called the primal form; where the Lagrangian dual form is formulated in this fashion:

$$(\omega^*, \mu^*) = \arg\max_{\mu} \inf_{\omega} \mathcal{L}(\omega, \mu) \tag{9a}$$

subject to

$$\mu_i \geq 0 \qquad \forall i. \tag{9b}$$

Because the primal form is convex and Slater's condition holds, we say the *strong duality* holds for our problem (Boyd and Vandenberghe 2004, Sect. 5.2.3). This means that the optimal value for the dual problem equals the optimal value for the primal form. As a consequence, solving the Lagrangian dual form (Eq. 9) is equivalent to solving the primal form of the problem (Eq. 7).

As stated, strong duality is obtained for our problem, and also the objective function is differentiable. Regarding these two criteria, an optimal value must satisfy the *Karush Kuhn Tucker (KKT)* conditions (Boyd and Vandenberghe 2004, Sect. 5.5.3). Here, we just exploit the *stationary* condition of KKT which states:

$$\text{for each component } j, \; \frac{\partial \mathcal{L}}{\partial \omega_j} = 0, \quad \text{i.e.,} \, \omega = \sum_{i=1}^{n} \mu_i \lambda_i. \tag{10}$$

We substitute equality constraint (10) into Eq. (8) which leads to a remodeled version of Eq. (9), in the form that we are interested in:

$$\mu^* = \arg\max_{\mu} \mathcal{L} = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \mu_j \lambda_i \cdot \lambda_j \tag{11a}$$

subject to

$$\mu_i \geq 0 \qquad \forall i. \tag{11b}$$

At this stage, we have just a simple positivity condition on μ_i which is more manageable than Eq. (7b) and more importantly, the preferences points appear just in the form of dot products $(\lambda_i \cdot \lambda_j)$.

Solving this new form of the problem gives optimal values for the Lagrange multipliers (μ^*); we then can use Eq. (10) to find ω^* as well.

3.2 Handling Inconsistencies

So far, we assumed that there exists at least one hyperplane such that all preferences points are placed in the positive open half-space of it. However, it is plausible that in practice this assumption may result in finding no feasible solution. To handle inconsistencies, we develop the basic model further by reformulating the initial constraint (7b) such that it could be violated with an appropriate amount of cost.

For this purpose, the constraint (7b) is rewritten as $1 - \omega \cdot \lambda_i \leq \xi_i$ where ξ_i represents the cost of violating the ith constraint and obviously should be non-negative $(\xi_i \geq 0)$. Then, the objective function is augmented by the term $C \sum_{i=1}^{n} \xi_i$ to reveal the effect of costs, where C is a constant parameter to be chosen by the user. The parameter C scales the impact of inconsistent points; a larger C corresponds to assigning a higher penalty to errors. Thus, the primal form of the problem becomes the following:

$$\omega^* = \arg\min_{\omega, \xi_i} \frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^{n} \xi_i \tag{12a}$$

subject to

$$1 - \omega \cdot \lambda_i - \xi_i \leq 0 \qquad \forall i, \tag{12b}$$

$$\xi_i \geq 0 \qquad \forall i. \tag{12c}$$

This optimisation problem (Eq. 12) is equivalent to *Optimisation Problem 1* (Ranking SVM) proposed in (Joachims 2002).

Because we have a new set of constraints $(\xi_i \geq 0)$, a new set of positive Lagrange multipliers $\alpha_i (i = 1, ..., n)$ are introduced. The Lagrangian changes as follows:

$$\mathcal{L} = \frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \mu_i \ (1 - \omega \cdot \lambda_i - \xi_i) - \sum_{i=1}^{n} \alpha_i \, \xi_i \tag{13}$$

The stationary KKT condition entails the new equality constraints:

$$\frac{\partial \mathcal{L}}{\partial \, \xi_i} = 0 \quad \text{i.e.,} \ \alpha_i = C - \mu_i \qquad \forall i. \tag{14}$$

By substituting $C - \mu_i$ for α_i in Eq. (13), the Lagrangian dual form of the problem becomes:

$$(\omega^*, \mu^*) = \arg\max_{\mu} \inf_{\omega} \mathcal{L} = \frac{1}{2}\|\omega\|^2 + \sum_{i=1}^{n} \mu_i \, (1 - \omega \cdot \lambda_i) \tag{15a}$$

subject to

$$\mu_i \geq 0 \qquad \forall i, \tag{15b}$$

$$\alpha_i \geq 0 \quad \text{i.e., } C - \mu_i \geq 0 \qquad \forall i. \tag{15c}$$

As in the previous section (Eq. 10), replacing ω with $\sum_{i=1}^{n} \mu_i \lambda_i$ gives us the simpler form of the problem:

$$\mu^* = \arg\max_{\mu_i} \mathcal{L} = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \mu_j \lambda_i \cdot \lambda_j \tag{16a}$$

subject to

$$0 \leq \mu_i \leq C \qquad \forall i. \tag{16b}$$

Since neither the ξ_i nor its associated Lagrange multiplier (α_i) appear in the objective function, this format of the problem is very similar to the one without introducing costs (Eq. 11), except μ_i is restricted by an upper bound which is C.

3.3 Non-linear Scoring Function

We assumed that the scoring function is a linear function. However, a non-linear scoring function might be a better choice sometimes. In this section, we deal with this matter to cover the non-linear case as well.

The *Kernel function* concept (Aiserman et al. 1964) is a widely-used trick for pattern recognition problems which can be also exploited for our case. The idea comes from the fact that a set of non linearly-representable data could be linearly managed if they mapped to a higher dimension. To do the mapping, we assume a function Φ of the form:

$$\Phi : \mathbb{R}^d \rightarrow \mathcal{H}, \tag{17}$$

where $\mathcal{H}$ denotes a higher dimensional space than d-dimensional. If all preferences points are mapped into $\mathcal{H}$ by making use of Φ, the Lagrangian function will be:

$$\mathcal{L} = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \mu_j \Phi(\lambda_i) \cdot \Phi(\lambda_j). \tag{18}$$

A deeper look into the process of computing $\mathcal{L}$ (Eq. 18) reveals that, even though $\mathcal{L}$ is still a scalar value, the optimisation problem performs a dot product operation $(\Phi(\lambda_i) \cdot \Phi(\lambda_j))$ in the high dimensional space which is computationally expensive.

Principally, a kernel is a function which operates in the lower dimension ($K(X,Y) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$), but yields an identical result to the dot product of mapped vectors in the higher dimension ($K(X,Y) = \Phi(X) \cdot \Phi(Y)$).

Due to the above property of the kernel function, the term $\Phi(\lambda_i) \cdot \Phi(\lambda_j)$ can be simply replaced in Eq. (18) with an appropriate kernel. The great advantage of such a replacement is that the complexity of the optimisation problem remains only dependent on the dimensionality of d (the preferences points space) and not of $\mathcal{H}$.

Note that this simplification happens without even explicitly stating the Φ function because the problem has been formulated in terms of dot products of points, the property that we insisted to have in the previous sections. So, the problem is rewritten as follows:

$$\mu^* = \arg\max_{\mu} \mathcal{L} = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \mu_j K(\lambda_i, \lambda_j) \tag{19a}$$

subject to

$$0 \le \mu_i \le C \qquad \forall i. \tag{19b}$$

There still remains one point that should be considered. Solving this optimisation problem only gives μ^* where after all, ω^* is required. Equation (10) cannot be used to attain ω anymore, because after the mapping, ω will live in $\mathcal{H}$; that is:

$$\omega = \sum_{i=1}^{n} \mu_i \Phi(\lambda_{\mathbf{i}}) \tag{20}$$

However, recall that finding ω^* is just an intermediate goal to achieve the scoring function. Therefore, using Eq. (20) brings the following form of the scoring function:

$$\begin{aligned} f(\mathbf{X}) &= \omega \cdot \Phi(\mathbf{X}) \\ &= (\sum_{i=1}^{n} \mu_i \Phi(\lambda_{\mathbf{i}})) \cdot \Phi(\mathbf{X}) \\ &= \sum_{i=1}^{n} \mu_i (\Phi(\lambda_{\mathbf{i}}) \cdot \Phi(\mathbf{X})) \end{aligned}$$

Importantly, this result contains points only in the form of dot products; incorporating a kernel function gives:

$$f(\mathbf{X}) = \sum_{i=1}^{n} \mu_i \ K(\lambda_{\mathbf{i}}, \mathbf{X}) \tag{21}$$

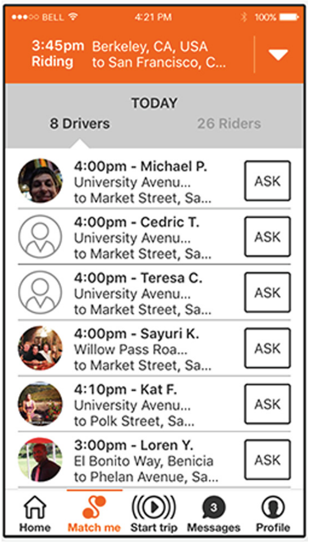
Equations (19) and (21) form the method, which is referred to as *SVPL* (Support Vector Preferences Learner) in our experiments. As seen, the input parameters that should be chosen in SVPL are C and the kernel function's parameters (if it has any).

4 Experiments

4.1 Data Repository

The experiments make use of a subset of a year's worth of real ridesharing records (from April 2014 to April 2015), provided by Carma[1] (formerly Avego). Carma is a software company currently offering a dynamic ride-share application for internet-enabled mobile phones (see Fig. 4). In addition to their App, any individual can create his own App, powered by the Carma's free API which uses Carma's live data.

The process of Carma's application is briefly explained so as to facilitate the understanding of the structure of the provided database. Users can request a ride-share as a driver, rider or both (happy to be rider or driver). Once a ridesharing request is created, the user can ask the system to find appropriate matching opportunities. Then, a suggestion list is shown on the user's screen. For each suggested item, the user can accept, decline or just leave it with no response. Note that accepting an opportunity at this stage does not mean the ride-share will certainly happen, because it also needs the confirmation of the other party; so, it is observed in the database that for a single trip a user may have several accepted items. While the system is waiting to get the second confirmation, the user can cancel an initially accepted item. It should be pointed out that the cancellation of a specific item differs from the cancellation of the whole trip request; the former often happens because of the greater attractiveness of a new



3:45pm Riding
Berkeley, CA, USA
to San Francisco, C...

4:00pm - Michael P.
University Avenu...
to Market Street, Sa...

4:00pm - Cedric T.
University Avenu...
to Market Street, Sa...

4:00pm - Teresa C.
University Avenu...
to Market Street, Sa...

4:00pm - Sayuri K.
Willow Pass Roa...
to Market Street, Sa...

4:10pm - Kat F.
University Avenu...
to Polk Street, Sa...

3:00pm - Loren Y.
El Bonito Way, Benicia
to Phelan Avenue, Sa...

Fig. 4. A screenshot of Carma's application to offer a ride-match.

---

[1] https://www.gocarma.com/.

item, and the latter is for the time when the user does not want to share the ride for the created trip anymore.

According to the above mechanism, a *ride-matching record* is associated with a class label which is among these four: *Accepted*, *Cancelled*, *Ignored* or *Declined*. As explained earlier, a class label comes from the response of the *main* user towards sharing the ride with the *target* user.

The second element of a ride-matching record is a vector of features, built from the personal information of both the main and the target user and the properties of their trips. Before starting the learning process, normalizing the features' spaces is an essential requirement due to the fact that margin based methods are known to be sensitive to the way features are scaled (Ben-Hur and Weston 2010). Because of that, features are scaled to an identical range, i.e., $[0, 1]$. The extracted features from the provided database are listed here:

- Positional Component: Expressing how suitable the pick-up and drop-off locations will be for the main user.
- Temporal Component: Expressing how appropriate the departure time will be for the main user.
- Gender Component: Indicating whether the target user is of the same gender (1) or the opposite (0).
- Has Image: It considers whether the target user's profile contains his/her picture or not.
- Average Rating: At the end of a ridesharing experience, users can rate the other party. This feature holds the average rate that the target user has gained from previous trips.
- Is Favourite: As well as the rating, a user can mark a particular user as a favourite person at the end of ridesharing. This feature shows whether the target user is among individuals who are marked as a favourite of the main user or not.
- Previous Rating: If the main user and the target user have had a previous ridesharing experience with each other, this feature shows the rating that the main user has given to the target user.

We base our experiments on 12 benchmarks derived from this data-set. Each benchmark corresponds to a different user (who is the main user of all ride-matching records of that benchmark). Table 2 shows the number of ride-matching records, separated by class labels, for each benchmark.

4.2 Experiments Settings

For a benchmark, the ride-matching records are sorted in ascending order of the creation time, and then split into two parts. The first part includes 80% of the records which we use to derive a set of pairwise comparisons as explained in Sect. 2.1. This set works as input data to train SVPL. At the end of the learning stage, SVPL can predict, for each record, a scalar value which expresses the goodness of that record for the user.

Table 2. 12 benchmarks derived from a commercial ridesharing system which are being used in the experiments.

Benchmark	Accepted	Cancelled	Ignored	Declined	Total
1.	179	72	83	14	348
2.	34	1	49	0	84
3.	210	52	138	2	402
4.	3	1	124	41	169
5.	81	1	53	29	164
6.	126	3	74	45	248
7.	39	7	28	4	78
8.	45	7	78	29	159
9.	146	37	3	1	187
10.	126	24	14	1	165
11.	40	1	41	13	95
12.	96	12	25	4	137
Total	1125	218	710	183	2236

The second part of the data is utilized for the testing stage. From this data, a weak order between the records (a kind of ranking in which incomparable items are assigned with the same rank) with respect to class labels could be derived (see Example 3 below). This ranking is used as the ground truth for testing. On the other hand, the predicted scalar values for records produced by the model suggest a total order (ordinal ranking) between records.

Thereafter, the *C-Index* (or concordance C) measure is exploited so as to assess the ranking performance of SVPL with respect to the ground truth ranking (Gönen and Heller 2005). It is calculated as follows:

$$\text{C-Index}(r, \hat{r}) = \frac{\kappa}{\kappa + \bar{\kappa}}$$

where κ is the number of correctly ranked pairs and $\bar{\kappa}$ (Kendall tau distance) is the number of incorrectly ranked pairs.

DCG is another metric that is used to evaluate the model's performance (Järvelin and Kekäläinen 2002). For a particular ranking among p items, DCG is defined as follows[2]:

$$\text{DCG}_p = \frac{rel_1}{0.4} + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i)}$$

where rel_i is the relevance degree of the item at position i. Here, we say the relevance of an accepted ride-matching is 3; a cancelled one is 2; ignored is 1 and

---

[2] The original formula is $\text{DCG}_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i)}$. It can be seen that the coefficient of the first term (rel_1) and the second term's (rel_2) are equal. To give a smaller penalty value to the first term, we divided rel_1 by 0.4 (instead of 1).

declined is 0. Since DCG_p also depends on p, it should be normalized. Thus, we use normalized DCG (nDCG) which is calculated by dividing "DCG of the current ranking" by "DCG of the ideal ranking".

In terms of choosing the kernel function, although there are many kernels available in the literature, and devising a new kernel by meeting *Mercer's Condition* is possible, we just simply utilize three well-known kernels, listed here:

- $K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$: The linear kernel which is equivalent to the non-kernel based formulation (Eq. 16).
- $K(\mathbf{x}, \mathbf{y}) = (\gamma \mathbf{x} \cdot \mathbf{y} + r)^p$: The polynomial kernel of degree p; here, we assume $p = 2$, $\gamma = 1$ and $r = 0$.
- $K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2}$: This is the Gaussian radial basis function (RBF). The parameter σ is tuned as an input parameter.

For adjusting the input parameters, we use the *Grid Search* algorithm for hyper-parameter optimisation (Bergstra and Bengio 2012).

Each run of SVPL, which comprises learning and testing phases for a benchmark, takes less than a couple of minutes, making use of a computer facilitated by a Core i7 2.60 GHz processor and 8 GB RAM memory.

4.3 SVPL and Maximising Saved Travel Distance

We stated in Sect. 2 that c_{ij} is a numeric representation to show how good the matching is between individuals i and j, which prevalently considers only the benefits of the whole system such as the total saved travel distance. However, we argued that this is insufficient and c_{ij} should also take into account user preferences. Then, we suggested making use of this measure to rank and recommend best ridesharing opportunities to the user. We claim this strategy would boost the users' satisfaction degree, which should increase the chances of repeat usage of the system.

Now, we examine the importance of the user preferences by comparing the effectiveness of ranking provided by SVPL, which involves learning user preferences, with the case when options are ranked regardless of the user's desires; i.e., an option A precedes another option B when the benefits obtained by that option for the whole system are more. Specifically, in our experiments, we consider the saved travel distance by a matching as the benefit gained for the system from that particular matching. The following example clarifies the situation.

Example 3. Suppose there were 6 ride-matching opportunities for Alice's trip D_1 as stated in Table 3. From her responses (i.e., accepted, cancelled, ignored and declined) the following multipartite ranking can be derived as ground truth.

$$R_1, R_2 \succ R_3 \succ R_4, R_5 \succ R_6$$

Note that R_1 with R_2, and R_4 with R_5 are incomparable. This is the ideal ranking of available options for this trip and its DCG is $({}^{3}\!/_{0.4} + {}^{3}\!/_{\log_2 2} + {}^{2}\!/_{\log_2 3} + {}^{1}\!/_{\log_2 4} + {}^{1}\!/_{\log_2 5} + {}^{0}\!/_{\log_2 6} =)$ 12.69.

Table 3. Six ride-matching opportunities available for Alice's trip D_1, which are being used in Example 3.

	Ground truth		SVPL prediction		Saved travel distance	
	Labels	Ranking	Value	Ranking	Value(km)	Ranking
R_1	A	1	8.2	3	32	6
R_2	A	1	**10**	1	36	3
R_3	C	3	8.5	2	35	4
R_4	I	4	6.9	6	38	2
R_5	I	4	7.5	4	**40**	1
R_6	D	6	7	5	34	5

Assume that SVPL has been previously trained by some ridesharing records of Alice, and now predicts those values in 4th column. As a result, SVPL ranks options in this fashion:

$$R_2 \succ R_3 \succ R_1 \succ R_5 \succ R_6 \succ R_4$$

Given this ranking, DCG is $(3/0.4 + 2/\log_2 2 + 3/\log_2 3 + 1/\log_2 4 + 0/\log_2 5 + 1/\log_2 6 =)$ 12.28 and subsequently nDCG $= 12.28/12.69 = 0.9677$. With respect to C-Index metric, the ranking accuracy is 84.61%, since only two pairs $((R_1, R_3)$ and $(R_4, R_6))$ out of 13 pairs were ranked incorrectly and $11/13 = 0.8461$.

On the other hand, we sort objects according to the travel distance that would be saved by each matching, which leads to this order of objects:

$$R_5 \succ R_4 \succ R_2 \succ R_3 \succ R_6 \succ R_1$$

It should be noted again that for proposing this ranking, we set aside Alice's preferences which might be implicitly stated in her past ridesharing records; therefore, unlike SVPL, there is no need to have a learning phase. For this instance of ranking, nDCG and C-Index are respectively 0.5959 and 0.3846, which both are smaller than SVPL's. From this point onwards, we call this strategy of ranking ridesharing opportunities (regarding saved travel distance) as *STD* for the sake of brevity.

In this example, if the top recommended option suggested by SVPL (R_2) is chosen by Alice to form a ride-share for D_1, then the saved travel distance will be 36 km which is 4 km less than the optimal matching (R_5). Thus, roughly speaking, we can say by use of SVPL strategy in ranking rather than STD, the saved travel distance would probably decline by 10%, compared to the optimal case, in this example. □

Now, a reasonable question is how much do we gain, in terms of ranking accuracy, and how much do we lose, in terms of total saved travel distance, if SVPL is exploited instead of STD for the real ridesharing data? Table 4 addresses this question.

Table 4. These experimental results aim to compare SVPL against STD, based on ranking accuracy (first four columns) and the saved travel distance.

Benchmark	nDCG		C-Index		Saved travel distance (km)		
	STD	SVPL	STD	SVPL	STD	SVPL	Loss rate (%)
1.	80.51	**94.81**	39.10	**70.73**	1123	1077	4.1
2.	62.08	**100**	25.00	**100**	248	178	28.4
3.	**98.35**	85.14	94.77	**95.59**	926	881	4.9
4.	97.37	**98.73**	66.67	**76.39**	240	236	1.6
5.	57.37	**97.00**	43.48	**63.77**	215	213	0.7
6.	56.10	**95.05**	61.01	**74.51**	422	414	2.0
7.	53.58	**98.57**	40.00	**80.00**	186	186	0
8.	80.95	**93.62**	**66.50**	58.67	343	332	3.2
9.	89.59	**96.52**	41.67	**87.50**	47	46	2.4
10.	93.49	**97.37**	44.44	**83.33**	397	384	3.1
11.	61.19	**99.87**	32.73	**94.55**	154	146	4.9
12.	72.90	**66.77**	77.42	**74.19**	167	159	3.8
Weighted avg.	79.42	**92.62**	58.08	**79.66**	512	491	4.3

In Table 4, the first four columns show the accuracy of models with respect to two different metrics, namely nDCG and C-Index. The greater number for each measure has been emboldened for easier comparison. As expected, SVPL beats STD in ranking accuracy for most of the benchmarks. On the other hand, apart from the second benchmark, the loss rate of saved travel distance by using SVPL is less than 5%. The last row illustrates the weighted average of rows, where the weight for each row is the number of records for that benchmark (last column in Table 2); i.e., a benchmark with a greater number of records has proportionally greater impact on the average.

4.4 SVPL and the Worst Point Model

The results in Table 4 might be conclusive enough to lead us to believe that considering user preferences in ranking ride-matching items will increase the accuracy. However, are the preferences of users in ridesharing sufficiently diverse that we should learn a specific model for each user? In other words, what if an ad-hoc ranking strategy, which is generally sensible for all users, is adopted? In this situation, it is presumed all users have relatively similar preferences.

To examine this question, we assess SVPL against a somewhat intuitive way of ranking which is called *Worst Point Model* (WPM). In WPM, the matching trip all of whose features are 0, is assumed as the worst match. This assumption makes some sense, since it means that the worst matching is when the positional component, temporal component, *is favourable*, *has image* and so forth, all are 0. Afterwards, the score of each item is found from its Euclidean distance from

Table 5. The ranking performance of SVPL and WPM are shown for two metrics, nDCG and C-Index.

Benchmark	nDCG		C-Index	
	WPM	SVPL	WPM	SVPL
1.	92.86	**94.81**	59.94	**70.73**
2.	80.12	**100**	62.50	**100**
3.	74.61	**85.14**	83.39	**95.59**
4.	90.90	**98.73**	45.14	**76.39**
5.	63.80	**97.00**	39.86	**63.77**
6.	94.60	**95.05**	73.01	**74.51**
7.	34.29	**98.57**	00.00	**80.00**
8.	93.62	93.62	58.67	58.67
9.	96.51	**96.52**	87.50	87.50
10.	94.41	**97.37**	55.55	**83.33**
11.	60.96	**99.87**	27.27	**94.55**
12.	**67.24**	66.77	35.48	**74.19**
Weighted avg.	82.52	**92.62**	60.02	**79.66**

the worst point; i.e., a higher score is produced by going away from the worst point. Needless to say no learning process is involved in acquiring these scores. Therefore, a ranking between ridesharing opportunities, based on their distance from the worst point, will be suggested. Table 5 shows that SVPL yields a higher performance than WPM for most of benchmarks.

4.5 SVPL and Regression Models

So far, our experiments have shown that learning user preferences by way of SVPL is an auspicious direction in ride-matching. However, as explained in Sect. 2.1, using a regression based method is another viable approach for learning weights. Recall that SVPL and the regression method require different formats for the input data, though they both eventually produce a scoring function. For the regression method, the main user responses (class labels) are converted to scalar values, and for SVPL, a set of pairwise comparisons among ride-matching records is created (Sect. 2.1 illustrates how).

Thus, in this subsection, the performance of SVPL is compared with three regression methods in Table 6. These rival methods are *Linear Regression*, *Neural Network Regression* and *Support Vector Machine Regression* which are respectively abbreviated as LR, NNR and SVM.

As seen in Table 6, SVPL outperforms the regression methods overall, which could evidently indicate that pairwise comparisons gives a more natural representation of user preferences rather than simply converting labels to numeric values.

Table 6. The ranking accuracy comparison between SVPL and three regression methods, namely linear regression (LR), neural network regression (NNR) and support vector machine regression (SVM).

Benchmark	nDCG				C-Index			
	LR	NNR	SVM	SVPL	LR	NNR	SVM	SVPL
1.	86.47	85.81	85.98	**94.81**	68.38	51.07	58.23	**70.73**
2.	100	100	100	100	100	100	100	100
3.	83.15	77.56	83.55	**85.14**	71.46	80.02	82.54	**95.59**
4.	98.54	**99.22**	93.65	98.73	74.31	**84.72**	48.61	76.39
5.	70.29	93.70	55.85	**97.00**	61.59	**72.46**	25.36	63.77
6.	70.12	69.88	67.37	**95.05**	67.01	67.01	53.52	**74.51**
7.	94.52	45.24	45.24	**98.57**	60.00	40.00	40.00	**80.00**
8.	**95.14**	91.54	83.55	93.62	65.17	57.67	56.83	**58.67**
9.	90.45	90.45	**96.52**	**96.52**	37.50	37.50	**87.50**	**87.50**
10.	92.11	94.74	94.74	**97.37**	50.00	66.67	66.67	**83.33**
11.	77.83	99.15	77.83	**99.87**	89.09	78.18	89.09	**94.55**
12.	61.52	61.04	66.49	**66.77**	48.39	29.03	70.97	**74.19**
Weighted avg.	84.04	83.77	80.77	**92.62**	65.11	63.89	64.93	**79.66**

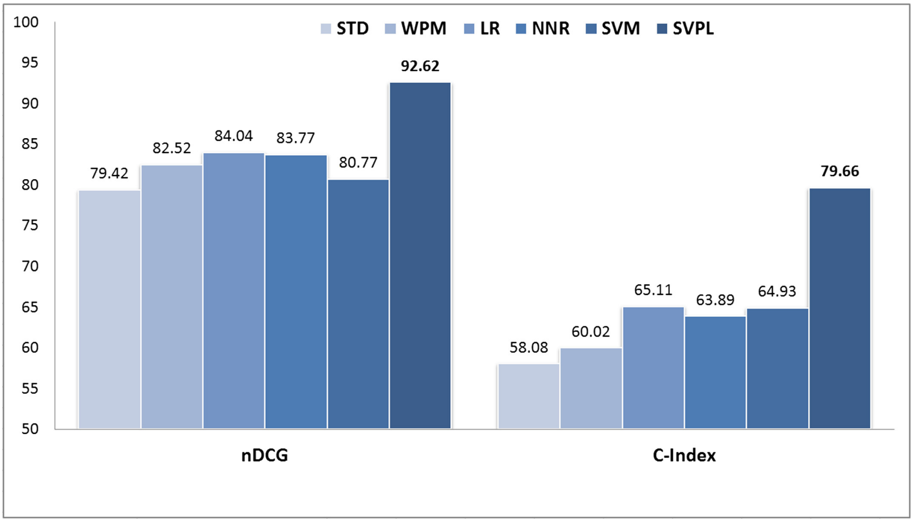


Fig. 5. The overall accuracy of models according to nDCG and C-index metrics. The models compared are: ranking based on saved travel distance (STD), Worst Point Model (WPM), Linear Regression (LR), Neural Network Regression (NNR), Support Vector Machine Regression (SVM) and Support Vector Preferences Learner (SVPL).

Finally, Fig. 5 gives an overview of our experimental results, where the weighted average of accuracy of each model (last columns in tables) have been drawn as a bar chart. The results show a clear superiority of SVPL over the other methods for the dataset.

5 Concluding Remarks

The advent of advanced technologies including GPS, web, and mobile technologies for real-time communication provides a unique opportunity to form new dynamic ridesharing systems, which could potentially provide huge societal and environmental benefits. At the heart of ridesharing concept, matching drivers and riders in real-time is prominently featured as a challenge.

In this paper, we have presented novel aspects of the automatic ride-matching system, making use of user preferences. We believe that a good understanding of participant behaviour and preferences will be essential; if ride-share matches do not satisfy participant preferences, the participant may not make use of the ride-share system in the future. Moreover, learning user preferences softens the traditional way of setting hard constraints, and removes the need for eliciting some complex parameters from participants.

Unlike the prevalent systems in ridesharing, our approach gives freedom to participants in the choice of ridesharing partners; it is supported by the fact that the user may not approve of the assigned matching found from the optimisation of the whole system. Of course, providing the flexibility in the matching process may cause deviation from the optimal solution with respect to overall saved travel time. However, the first part of our experiments indicated that this loss could be quite small.

SVPL as a model with the ability to learn user preferences in a natural way, was discussed in Sect. 3. Our experimental results indicated the effectiveness of SVPL, in comparison with five competitor strategies. The intention is that SVPL could contribute in the suggestion of a menu of good choices to the user in a ridesharing system.

A natural extension would be if the preferences set (pairwise comparisons) derived from the user's feedback, could be associated with some degree of uncertainty. For instance, if it is said that the user probably prefers the case A to B, with certainty degree of 0.7. Other interesting subjects related to this paper that could be addressed in future works, include handling nominal and unknown values in the features space, and online learning.

Acknowledgment. This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289, and also supported by Carma (http://carmacarpool.com). We are grateful for helpful discussions with Vincent Armant, Cristina Cornelio, Martin Gerner, Barry O'Sullivan, Conor Roche and Gilles Simonin.

References

Agatz, N., Erera, A., Savelsbergh, M., Wang, X.: Optimization for dynamic ride-sharing: a review. Eur. J. Oper. Res. **223**(2), 295–303 (2012)

Agatz, N.A., Erera, A.L., Savelsbergh, M.W., Wang, X.: Dynamic ride-sharing: a simulation study in metro atlanta. Transp. Res. Part B Methodol. **45**(9), 1450–1464 (2011)

Aiserman, M., Braverman, E., Rozonoer, L.: Theoretical foundations of the potential function method in pattern recognition. Avtomat. i Telemeh **25**, 917–936 (1964)

Amey, A., Attanucci, J., Mishalani, R.: Real-time ridesharing - the opportunities and challenges of utilizing mobile phone technology to improve rideshare services. Transp. Res. Rec. J. Transp. Res. Board **2217**(1), 103–110 (2011)

Baldacci, R., Maniezzo, V., Mingozzi, A.: An exact method for the car pooling problem based on Lagrangean column generation. Oper. Res. **52**(3), 422–439 (2004)

Ben-Hur, A., Weston, J.: A User's Guide to Support Vector Machines, pp. 223–239. Humana Press, Totowa (2010)

Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(1), 281–305 (2012)

Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)

Burges, C.J.: A tutorial on support vector machines for pattern recognition. Data Mining Knowl. Discov. **2**(2), 121–167 (1998)

Calvo, R.W., de Luigi, F., Haastrup, P., Maniezzo, V.: A distributed geographic information system for the daily car pooling problem. Comput. Oper. Res. **31**(13), 2263–2278 (2004)

Chan, N.D., Shaheen, S.A.: Ridesharing in North America: past, present, and future. Transp. Rev. **32**(1), 93–112 (2012)

Chaube, V., Kavanaugh, A.L., Perez-Quinones, M.A.: Leveraging social networks to embed trust in rideshare programs. In: 2010 43rd Hawaii International Conference on System Sciences (HICSS), pp. 1–8. IEEE (2010)

Duggan, M., Smith, A.: Cell internet use 2013. Pew Research Center (2013). http://www.pewinternet.org/2013/09/16/cell-internet-use-2013/

Emarketer. 2 billion consumers worldwide to get smart(phones) by 2016 (2016). http://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694

Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M.-E., Wang, X., Koenig, S.: Ridesharing: the state-of-the-art and future directions. Transp. Res. Part B Methodol. **57**, 28–46 (2013)

Ghoseiri, K., Haghani, A.E., Hamedi, M.: Real-time rideshare matching problem (2011)

Gönen, M., Heller, G.: Concordance probability and discriminatory power in proportional hazards regression. Biometrika **92**(4), 965–970 (2005)

Herbawi, W.M., Weber, M.: A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time windows. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation. ACM, pp. 385–392 (2012)

Hwang, M., Kemp, J., Lerner-Lam, E., Neuerburg, N., Okunieff, P.: Advanced public transportation systems: state of the art update 2006. Technical report (2006)

Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. (TOIS) **20**(4), 422–446 (2002)

Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002, pp. 133–142. ACM, New York (2002)

Saranow, J.: Carpooling for grown-ups—high gas prices, new services give ride-sharing a boost; rating your fellow rider. Wall Street J. 2 Feb 2006. https://www.wsj.com/articles/SB113884611734062840

Smith, A.: US smartphone use in 2015. Pew Research Center (2015). http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/

Winter, S., Nittel, S.: Ad hoc shared-ride trip planning by mobile geosensor networks. Int. J. Geog. Inf. Sci. **20**(8), 899–916 (2006)

Zickuhr, K.: Three-quarters of smartphone owners use location-based services. Pew Research Center (2012). http://www.pewinternet.org/2012/05/11/three-quarters-of-smartphone-owners-use-location-based-services/

An Altruistic-Based Utility Function for Group Recommendation

Silvia Rossi[1(✉)], Francesco Cervone[1], and Francesco Barile[2]

[1] Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione,
University of Naples Federico II, Via Claudio, 80125 Napoli, Italy
`silvia.rossi@unina.it`
[2] Dipartimento di Matematica e Applicazioni, University of Naples Federico II,
Via Cintia, 80126 Napoli, Italy
`francesco.barile@unina.it`

Abstract. Preference aggregation strategies, that are inspired by economic models of decision makers, typically assume that the individual preferences of the group members depend only on their own individual evaluations of the considered items. In this direction, group recommendation algorithms rely on such standard aggregation techniques that do not consider the possibility of evaluating social interactions and influences among group's members, as well as their personalities, which are, indeed, crucial factors in the group's decision-making process, especially regarding small groups. On the contrary, the laboratory data have encouraged the development of models of other-regarding preferences since altruism, fairness, and reciprocity strongly motivate many people. In this paper, starting from a utility function from the literature, which combines the user personal evaluation of an item with the ones of the other group members, we propose a group recommendation method that takes into account altruism. Such function models the level of a user's altruistic behavior starting from his/her agreeableness personality trait. Once such utility values are evaluated, the goal is to recommend items that maximize the social welfare. Performance is evaluated with a pilot study and compared with respect to Least Misery. Results showed that while for groups of two people Least Misery performs slightly better, in the other cases the two methods are comparable.

Keywords: Social choice · Group recommendation
Five-factor model · Other-regarding preferences

1 Introduction

A customer buying decision-making process is a complex activity that involves different stages: the recognition of the need, the information search, the evaluation of the available alternatives and, finally, the purchase decision [13]. Online stores make the process easier by collecting a large amount of data to help the user in the search of a product. However, the user may suffer from choice overload since too many options have to be evaluated [45]. In order to avoid these

© Springer International Publishing AG, part of Springer Nature 2018
N. T. Nguyen et al. (Eds.): TCCI XXVIII, LNCS 10780, pp. 25–47, 2018.
https://doi.org/10.1007/978-3-319-78301-7_2

phenomena, Recommender Systems (or simply RSs) are used to provide suggestions about relevant items for users. The list of recommended items is the result of a process that aims to understand, for example, which product a specific user wants to buy, which music to listen, which movie to see or which news to read. Moreover, some domains are characterized by products that can be consumed while being in a group, such as, for example, a movie or a dinner with friends. In this context, the goal of Group Recommender Systems (GRSs), differently from individual RS, is to recommend items to a whole group of users taking into account all the individual preferences.

The study of GRSs is still an open research problem. Typically, preferences and tastes of individuals are collected in the same way of RSs, but the real problem falls into the correct aggregation of these preferences in order to recommend items [28]. The problem of aggregating individual preferences has been widely studied in Mathematics, Economics and Multi-Agent Systems, with the definition of Social Choice functions. In literature, several solutions have been proposed. The most common technique is the Average Satisfaction (AS) that treats all group members as being equal by averaging preferences (or utilities) of all the members in order to produce a final list for the entire group. Least Misery (LM), instead, cares about the possible dissatisfaction of some members by choosing items that minimize it. Since economic models traditionally assume that decision-makers' utility values represent only their own evaluations of the considered item, no one of the standard techniques considers that there are also other factors that can influence the group decision. However, groups can be dynamic, as well as members' behaviors depending on situations. Real *group decision making* is a complex mechanism that involves relationships among the members, users' personality, as well as their experience about the domain of interest. Recently, economic models of other-regarding preferences [9,12,18] have been proposed, which concerns for altruism, fairness, and reciprocity.

In this work, we study the influence of individual users' personality in the group decision-making process. We focus on small groups since small group decision making rely on mechanisms (e.g., interpersonal relationships, mutual influences and user personality) that are different with respect to the ones adopted for larger groups [26]. In particular, we focus on the personalities of the group members that, in a realistic scenario, can have a great impact on the group decisions [41]. For example, there can be people that rarely change their minds because they believe that their own decision is the best for everyone, or simply because they do not want to reduce their utility in favor of others. Other types of people instead, can be worried about the satisfaction of all the other members, at the cost of the personal one. This phenomenon, that is related on how people satisfaction is influenced by the satisfaction of other people, is known as *Emotional Contagion* [29], and the motivations, mechanism of working, and many other aspects, are still open research problems in the psychology field [17]. However, many empirical studies show the evidence of this influence between people [29]. Thus, the latter ones are willing to lose some utility in order to reach a valid and suitable agreement for the entire group. Furthermore, In order to consider these elements, it is necessary to study users' personalities through some models proposed in human sciences area. Among the many models of personality, the Five Factor Model (FFM) appears suitable for usage in recommender systems as it can be quantitatively measured (i.e. numerical values for each of the considered factors, namely, openness, conscientiousness, extraversion, agreeableness, and neuroticism) [47]. In particular, in this work, we focus on the role of the *agreeableness* factor in the definition of a parametric model of other-regarding preferences for altruistic behavior.

This paper is organized as follows. In Sect. 2, we introduce Group Recommendation open problems and we describe some related works. In Sect. 3, we introduce the FFM used to evaluate users' agreeableness personality trait values, while, in Sect. 4, we introduce the utility function used to model altruistic behaviors and the evaluation, by personality tests, of the parameters that characterize such function. In Sects. 5 and 6, we present the developed application and the metrics used to evaluate its performance both with offline testing and with real users with respect to Least Misery. Finally, conclusions are discussed in Sect. 7.

2 Background and Related Works

Generally, in order to design a group recommendation mechanism, two main approaches are used. Starting from the users' preferences (one for each of the group's member), a first approach relies on merging such preferences in order to obtain a single "virtual" profile for the whole group. Then, it uses a single user RS on this profile to generate the recommendations for the group. On the contrary, a second approach firstly generates the recommendations starting from single user's profiles, and then it merges these recommendations using some group decision strategy. In this case, we refer to *Social Choice* functions [3]. These functions, according to [46], can be classified as majority-based (mainly implemented as voting mechanisms to determine the most popular choices among alternatives), consensus-based (that try to average among all the possible choices and preferences), and role-based (that explicitly take into account possible roles and hierarchical relationships among members). Examples of these techniques are illustrated in [3,28], whereas the most common approaches are based on the average satisfaction and least misery techniques. While, according to [4], in some cases, preferences aggregation can lead to better results, the use of such virtual user profile for the group makes it difficult to provide additional explanations for the provided recommendations [23].

The results presented in the literature showed that there is no aggregation strategy can be defined as the "best", but different approaches are better suited for different scenarios, depending from the characteristics of the specific group [19]. Besides, such techniques do not seem to capture all the features of real-world scenarios. For example, automatic ranking mechanisms often require that all the agents involved have the same influence in the decision procedure, while real group interactions take into account intra-group roles [34], mutual influences and tie strength [36], as well as context dependent factors as the emotional contagious [29]. For example, some members of the group could have a particular influence on the others, based on their personal experiences in the considered domain [19] or on the strength of their mutual relationships [40]. Usually, the decision of a group member whether or not to accept a given recommendation may depend not only on his/her own evaluation of the content of the recommendation but also on his/her beliefs about the evaluations of the other group members [8]. Users may be willing to accept last preferred activities in order to improve the group happiness. Hence, recommendation systems for groups need to capture both preferences of the group members but also these key factors in the group decision process [39].

On the basis of these considerations, in literature, some approaches are starting to model GRSs that weights users preferences in different ways according to specific user-related parameters. For example, in [19] users preferences are weighted according to their expertise, while in [40] users' dominance and influence are taken into consideration in order to weigh more the preferences of dominant people in a group. In the proposed approach, we do not consider weights in the aggregation process, but it is the utility function, which is used to evaluate users rating on individual items, that takes into account the preferences of the other members of the group more or less with respect to the user agreeableness trait. In this sense, our work is related to the approach of [44], where individual empathetic utilities are defined taking into account local relationships with neighborhoods in a social network. Here, the authors introduce the concept of empathetic utility on social networks: the satisfaction of an individual depends from both his intrinsic utility and his empathetic utility deriving from the happiness of his neighbors in the network [44]. Based on this idea, individual preferences are aggregated in a weighted utility function that takes into account local relationships with neighborhoods in the network. However, in [44] the Authors do not specify how to evaluate such numerical relationships, while they focus on computational aspects of scaling up with large networks of friends. Finally, notice that multi-attribute utility functions, in recommender systems, are typically used to evaluate, for each user, an item rank that depends on the attribute values of the considered item [7]. Here, the parameters of the function are not the attributes of the items but the evaluations of the item rank by the considered user as well as by the other members of the group. The balance of these two components is the user personality in term of his/her agreeableness.

The role of personality has been addressed before, in literature, to improve the performance of RSs in the case of new users [22,33], since a number of studies showed that personality relates strongly to user preferences [38]. Moreover, it has been shown that people with different personalities can be more or less inclined to consume novel items, so the degree of diversity in presenting recommended items can be personalized accordingly [49]. However, to our knowledge, this is the first attempt to introduce personality factors in group decision making through the use of personality-based utility functions. The only relevant approach is the one of [36,37] and that of [42], where the personality of every individual in the group is evaluated in terms of conflicting resolution styles. Conflicting resolution styles are evaluated by using questionnaires, and, in particular, the Thomas-Kilmann Conflict Mode Instrument (TKI) [24], that is designed to measure the behavior of people in terms of levels of assertiveness or cooperativeness. Such two factors are related to the personality models of competitive, collaborative, avoiding, accommodating, and compromising traits. The user's ranking of an item is modified by considering all the couples of users and their value of assertiveness and cooperativeness in different ways (e.g., as multiplicative or additive factors). In our opinion, TKI models could be very useful in designing a simulated iterative decision making process that goes beyond the use of such factors as weights in aggregation functions. On the contrary, in this paper, we rely on the five-factor model and a utility function from the literature.

Finally, in [25], the authors model the group influences as a graph whereas every edge has a weight, which refers to the level of emotional contagion between two users. These are dependent on various personal aspects such as the personality, the social status or the relationship type. Four personal characteristics were considered, Competing (from TKI model) and Neuroticism, Extraversion and Cooperativeness (from NEO-FFI model), that should express user potential for influencing other group members. However, not enough results and justifications are provided to support their choices.

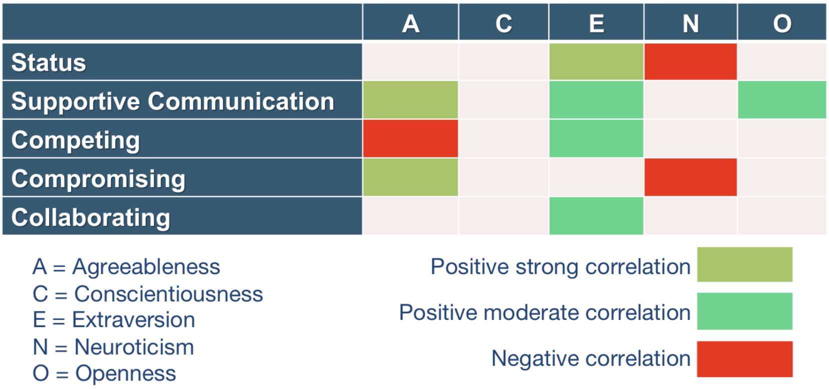
3 Big-Five in Decision Making

Research has shown that personality is a primary factor which influences human behaviors. Research in psychology has come out with different models to describe the human beings' personality. Among them, the most popular approach among psychologists for studying personality traits is the **Five-Factor Model** (FFM), which describes human personality using five factors (OCEAN), also known as **Big-Five** [31]. *Openness* represents the inclination to openness to new experiences, having an active imagination and a preference about the will to find new ideas [11]. Closed people are less flexible and rarely understand others' point of views. *Conscientiousness* describes how much an individual is responsible, disciplined and dutiful [32]. *Extraversion* is an indicator of assertiveness and trust. Extroverted people easily create interpersonal relationships and love working and being together with others [11]. *Agreeableness* describes the level of sympathy, availability, and cooperativeness. People with low level of this factor are competitive, skeptics and antagonistic. It measures how much a person is nice and altruistic [11]. Finally, *neuroticism* represents an emotional instability characterized by negative emotions like fear, anger, sadness and low self-esteem [32]. People with an high neuroticism trait rarely are able to control their impulses and cope with stress [30]. Among the many models of personality, the Five Factor Model (FFM) appears suitable for usage in recommender systems as it can be quantitatively measured [47].

Several studies tried to find correlation among personality traits and the group decision making process. For example, in [1], the authors tried to correlate

Big-Five with the *status*[1] of college students. A student with a high level of status implies a considerable level of attention by others towards him. Furthermore, status relates to respect and influence in the social context. Authors measured students' status through an assessment made by the student themselves and according to the positions that they play in the college organization. Results showed that the extraversion has a positive correlation with the status; conversely, neuroticism has a negative correlation, but only for men. The latter result can be explained because [6] said that sadness, depression, fear, shame and embarrassment (neuroticism features) are viewed as "unmanly". Thus, men who show these emotions are evaluated more negatively with respect to women. For the latter, these emotions are considered ordinary. In [43], the authors studied the impact of Big-Five in producing a *supportive communication*. The latter is opposed to *defensive communication* in which group members see others as a threat and try to stay one step ahead them. In *supportive communication* instead, all the members allow others to express their own opinions and consider other choices. Extraversion, agreeableness, and openness have a positive correlation in this context. In [27], the authors studied the influence of personality in conflict situations that happen during negotiations. He correlates *conflict resolution styles*, which represent different people reactions in case of conflict, with big-five by using two factors: the *assertiveness*, which describes how much effort an individual put in trying to satisfy his/her needs, and *cooperativeness*, which suggests the level of collaboration and the intention of maximizing others' utilities. Results showed that: neuroticism is negatively correlated to *compromising*; extraversion is positively correlated to *competing* and *collaborating*; agreeableness is negatively correlated to *competing* and positively to *compromising*; finally, conscientiousness and openness have no significant correlations.

The previous considerations are summarized in Fig. 1. Starting from those, in this work, we decided to move a first step by considering only a single



	A	C	E	N	O
Status			🟩	🟥	
Supportive Communication	🟩		🟢		🟢
Competing	🟥		🟢		
Compromising	🟩			🟥	
Collaborating			🟢		

A = Agreeableness
C = Conscientiousness
E = Extraversion
N = Neuroticism
O = Openness

Positive strong correlation 🟩
Positive moderate correlation 🟢
Negative correlation 🟥

Fig. 1. Correlation among personality traits and decision making in groups.

---

[1] Status describes the role that a person has in a social group.

personality trait, while leaving the combination with the others as a future work. In particular, we decided not to rely on the evaluation of neuroticism, which has negative correlations with status (a neurotic person could be excluded during a decision process) as well as with a compromising behavior (that, in contrast, may help the group decision process). Conversely, extraverts could be viewed as leaders, but at the same time they are more competitive than collaborative: all of those are qualities that are suitable to our case (the choice of a movie), but already considered in related works. A conscientious person could have a positive impact in the decision-making process, but we do not believe it is significant for our specific goal. This is the same for open individuals which could agree to see a movie not suitable with respect to their preferences, but whose trait could be more difficult to model with respect to the agreeableness factor.

We believe that in choosing a movie in a group of close friends, agreeableness (that is related to altruistic behavior [11] and negatively with respect to competition) plays an important role. It is obvious that, unless all the components have the same movie tastes, someone will have to give up their desires to see a movie which others do not like. Therefore, agreeable people will make compromises and, for this reason, we decided to analyze the impact of this factor in a GRS. People with high a value of agreeableness, just because of its descriptive characteristic, are more altruistic: basically, it means that this type of people care about the satisfaction of the entire group, or are more willing to compromise in order to obtain a solution that works for the whole group.

Once we identified a reasonable personality trait to consider, the second problem is to understand how to use such factor in computing the group's recommendation. With respect to other personality factors, such as the dominance or the leadership that could be modeled as weights in the aggregation process, in our opinion, the agreeableness cannot be treated simply as a weight. Instead of predicting an aggregated item rating for the entire group, the solution, we propose, is to assign to the item a single value for each user that takes into account the personal evaluation of the item and the ratings provided by the other members of the group. The level of agreeableness, in this case, will provide the balance between these two components. The function will be described in the following section.

4 Agreeableness and Other-Regarding Preferences

Generally speaking, the aim of an RS is to predict the relevance and the importance of items (for example movies, restaurants and so on) that the user never evaluated. More formally, given a set of n users ($G = \{1, \ldots, n\}$) and a set of m items ($M = \{1, \ldots, m\}$), an individual RS, for each user i, aims at building a *Preference Profile* of the user i over the complete set M ($\succ_i = \{x_{i,1}, \ldots, x_{i,m}\}$), with $x_{i,j} \in \mathcal{R}$, which represents the user i predicted evaluations for the j item, starting form some initial evaluations each user provides on some elements of M. Since our application domain is of movie recommendation, evaluations on items are typically expressed as rating values (ranging from 1 to 5).

Once each user $i \in G$ has a preference profile $\succ_i$ over M, the goal of a GRS is to obtain $\succ_G = \{x_{G,1}, \ldots, x_{G,m}\}$, where $x_{G,j}$ is the correspondent ranking for the j movie, as evaluated for the group. Typically, this is obtained by implementing a social choice function $SC : \succ^n \rightarrow \succ_G$, that aggregates all the preferences profiles in $\succ_G = \{x_{G,1}, \ldots, x_{G,m}\}$. Note that our goal is not to guess the exact value of $x_{G,j}$ the whole group would assign to the movie j, but to properly select the k-best movie in the group preference profile (the ones with the highest rating) and suggest them to the group.

Economic models traditionally assume that decision-makers are exclusively motivated by self-interest. Hence, utility values (and in this case item ratings) represent only their own evaluation of the considered item. Competitive markets, as well as many other contexts, are well simulated by this rationality assumption. However, in the last two decades, the laboratory and field data have encouraged the development of models of other-regarding preferences [9,12,18]. Those models suggest that the concern for altruism, fairness, and reciprocity strongly motivate many people, and so the evaluation of the personal utility, while in the presence of others, should take into account such factors. *Other-regarding preferences* that concern for the well-being of others, for fairness and for reciprocity, cannot be ignored in social interactions [18]. Moreover, the interaction between fair and selfish individuals is also a key to understanding the observed behavior in strategic settings [18].

In this work, we propose to consider the agreeableness factor in the process of building recommendations for groups. Instead of defining a specific ad-hoc social choice function that considers such factor, the proposed solution relies on the definition of an individual *utility function* (to build up the preference profile) to evaluate the rating of items for each user that takes into account the whole group. Such utility function could be interpreted as "the user satisfaction if the recommender system chooses that item for the group". So, once that the utilities for each item and each member of the group are computed, the considered social choice function simply selects the items that maximize, for example, the social welfare (i.e., the items that correspond to the highest sum of utilities).

For the utility function, among the recently developed models of other-regarding preferences, we used a model developed by Charness and Rabin in [9]. In particular, let us consider a group of n users, which have a rating for each element j of the domain (e.g., an expected rating provided by an individual RS). The utility function is composed of two terms; the first concerning a "disinterested" social-welfare criterion, defined as follows:

$$\begin{aligned} W(x_{1,j}, x_{2,j}, ..., x_{n,j}) = {} & \delta \cdot \min(x_{1,j}, x_{2,j}, ..., x_{n,j}) \\ & + (1-\delta) \cdot (x_{1,j} + x_{2,j} + ... + x_{n,j}) \end{aligned} \tag{1}$$

where $\delta \in [0,1]$, $x_{1,j}, x_{2,j}, ..., x_{n,j}$ are called *payoffs* and they are the individual rating of each of the n users of the j-th item. The aim of the first addend is reducing inequity (by helping the worst-off person); indeed, this value increases proportionally to the minimum payoff. This factor is, basically, a generalization of LM technique just because the utility is given by the minimum satisfaction among the users. The second addend is responsible for the maximization of the *social welfare* and increases proportionally to the sum of the individual payoffs. Setting $\delta = 1$, indeed, the function cares only about inequity, just like LM. Setting $\delta = 0$ instead, the function focuses on global satisfaction.

The utility of user i about the item j is a weighted sum of the disinterested social-welfare W and its own payoff defined as follows:

$$U_{i,j}(x_{1,j}, x_{2,j}, ..., x_{n,j}) = (1 - \lambda_i) \cdot x_{i,j} + \lambda_i \cdot W(x_{1,j}, x_{2,j}, ..., x_{n,j}) \tag{2}$$

where $\lambda_i \in [0, 1]$ means how much a person pursues its own interest or the social welfare. By setting $\lambda_i = 0$, the user takes care only of his personal interest. $\lambda_i = 1$ represents the classical disinterested behavior, of him/her who does not take part in the group decision, moving the weight of the function on the entire group satisfaction (including its own). Hence, according to [9], the considered function evaluates both the personal and the group satisfaction, depending on an altruistic factor λ_i. Once the new items utilities are evaluated, the goal is to recommend movies that maximize the *social welfare*.

Finally, notice that in the case of a group composed only of two people, Eq. 2 becomes:

$$U_{i,1}(x_{1,1}, x_{2,1}) = \begin{cases} x_{i,1} + \lambda_i(1 - \delta)x_{j,1} & \text{if } x_{i,1} < x_{j,1} \\ (1 - \lambda_i\delta)x_{i,1} + \lambda_i x_{j,1} & \text{if } x_{i,1} \geq x_{j,1} \end{cases} \tag{3}$$

Hence, the utility of a decision maker takes into account the well-being of the other person, but less so if the other person is better off than he/she is.

Since the agreeableness factor is positively correlated with an altruistic behavior, here, our goal is to calibrate the λ_i parameter with respect to the individual agreeableness levels, so that the most altruistic individuals have a high value and viceversa.

4.1 Personality Test

Personality can be acquired in both explicit and implicit ways [16]. The former measures a user's personality by asking the user to answer a list of designed personality questions. These personality evaluation questionnaires have been well established in the psychology field [21]. The implicit approaches acquire user information by observing users' behavioral patterns. Typically, explicit personality acquisition interface are preferred by the user [16]. However, implicit methods require less effort from users. In our study, we adopted the explicit way to measure users' personality, but we selected a questionnaire that requires only a small set of questions, and so a minimum required effort from the user.

There are several questionnaires to predict a person's Big-Five factors. Some of these consist of a lot of questions, in certain cases some hundreds. Too long questionnaires can reduce the people's attention level. A lot of research groups tried to cut the number of questions preserving the accuracy. The most famous small questionnaires are NEO-FFI (NEO Five Factor Inventory) composed of

Table 1. Mini IPIP questionnaire for agreeableness evaluation.

#	Text
1	Sympathize with others' feelings
2	Am not interested in other people's problems (R)
3	Feel others' emotions
4	Am not really interested in others (R)

(R) = Reverse Scored Item.

60 questions [10], the 50-item IPIP-FFM (International Personality Item Pool - Five Factor Model) [20], the 44-item BFI (Big Five Inventory), the TIPI (Ten-Item Personality Inventory) [21] and, finally, the **Mini-IPIP** (Mini International Personality Item Pool) [14]. We chose the last because it is very short, but at the same time effective. It consists of 20 questions, 4 for each personality factor. In Table 1, we reported the question concerning the evaluation of the user agreeableness (questions should be read in the first person). The answer to each question can be a number from 1 to 5, where 1 means "very inaccurate" and 5 "very accurate".

4.2 Parameter Settings

Let us assume that each user completed the personality evaluation test. A generic user i has a personal *agreeableness* value $\alpha_i \in [1, 5]$, obtained by averaging individual answer values. Since the λ_i parameter of Eq. 2 should belong to the range $[0, 1]$ and should depend directly on α_i, we defined it as follows:

$$\lambda_i = \frac{\alpha_i - 1}{4}. \tag{4}$$

This way, when $\alpha_i = 5$ (high level of *agreeableness*) $\lambda_i = 1$, so the user cares only about the group satisfaction. When $\alpha_i = 1$, instead, $\lambda_i = 0$ and, consequently, user's utility depends only on his personal satisfaction. As we already explained, in the Eq. 1 there is the δ parameter that deals with weighting inequity aversion against global satisfaction. In Sect. 5, we show an offline experiment to tune such parameters. Finally, the utility function of Eq. 2 refers to the utility of a particular item for a specific user. Thus, payoff $x_{1,j}, x_{2,j}, ..., x_{n,j}$ stand for the rating predictions of a generic item for the n members of the group.

4.3 Payoffs and Group Recommendations

One of the most important choices for an RS concerns the creation of the recommendation list. The ideal process would be to calculate the individual payoffs for all the items in the dataset. At the end of this process, we should aggregate the predictions and build up the recommendation list for the group.

Given that the datasets are composed of thousands of items, this solution would be computationally inefficient.

Some researchers remedied to this problem by restricting the number of considered items in the prediction process. In [35], for example, 15 movies are a priori chosen from the dataset, and the group must select the preferred three movies out of this set. Also in [48], a similar solution is used, but with a larger number of items. In [5], the Netflix platform functionality is exploited, so users can directly populate a list of movies they are interested in. Here, we decided to generate the group's recommendation only starting from the *k-best* movies for each user. Hence, firstly, the system creates a list of 10 items L_i for each user i evaluated by an RS. Later, it merges all the lists in a single one which we call L:

$$L = \bigcup_{i \in G} L_i \tag{5}$$

where, G is the set of members of the group.

Our GRS computes for each user the rating predictions of all the items in L. The latter are exactly the payoffs $x_{1,j}, x_{2,j}, ..., x_{n,j}$ of Eq. 2 (where $n = |G|$ and j is the movie). Values of $U_{i,j}$, for each user i and for each item j, are then computed. Finally, $x_{G,j}$ denotes the utility of the group if the GRS chooses item j that is defined as follows:

$$x_{G,j} = \sum_{i \in G} U_{i,j}. \tag{6}$$

Our goal is to maximize the *social welfare*, indeed, the system will recommend the 10 items with the highest $x_{G,j}$ value.

5 Offline Evaluation

In order to evaluate the impact of the λ and δ parameters in our utility function, and to properly set the δ value, we conducted an offline evaluation. Since there are no available dataset for GRS, we adopted an approach similar to that used in [2] for generating random groups of different sizes (from two to eight users) starting from the users in a dataset. Then, we produced the recommendation lists for the group as previously described.

5.1 Movie Recommendation Server

We adopted the Apache Mahout library[2] to predict the user's ratings and chosen the MovieTweetings [15] dataset to train the system and to populate the *Ratings Repository*. MovieTweetings consists of movie ratings contained in well-structured tweets on the *Twitter.com* social network, and is one of the most used datasets in literature since it contains real users evaluation on a flexible

---

[2] http://mahout.apache.org/.

domain, such is the movie one, that is monthly updated. It was also used for the challenge proposed in RecSys 2014 conference[3]. The information contained in the dataset are divided into three files: *users.dat*, *ratings.dat*, and *movies.dat*, which provide, respectively, the user identification number, his/her associated ratings and a list of movies. The dataset is updated every day, therefore, its size is constantly changing. At the last access, it contained about 35000 users, 360000 ratings, and 20000 movies.

The recommendation engine provides rating predictions when the recommendation API is invoked. To achieve this goal, we used *item-based City Block distance*, also known as *Manhattan distance.* In Mahout implementation, the generic movie j is represented by a boolean vector:

$$j = [x_{1,j}, x_{2,j}, ..., x_{k,j}], \tag{7}$$

where k is the number of users in the dataset and $x_{i,j} = 1$ if user i rated the movie j. The distance between two movies rated by user i is the sum of the absolute value of the differences of the two associated vector components. More formally, the distance between items j and h is:

$$d(j, h) = \sum_{i=1}^{k} |x_{i,j} - x_{i,h}|. \tag{8}$$

5.2 Results

We divide the dataset in *training set* (R_{train}) and *test set* (R_{test}). For each user u in the dataset, 70% of his/her ratings are in the first set, and 30% in the second one. For the offline evaluation only, we generated a list of 100 items for each user, and then, we merged them in a single list L of the *top*@k movies, where the @k parameter indicates the length of the recommendation list, which varies in the set $\{5, 10, 20\}$.

In order to evaluate the quality of the recommendations, we used two information retrieval standard metrics, the *Normalized Discounted Cumulative Gain* ($nDCG$) and $F1 - score$. The metrics are modified in relation to the size k of the recommendation list, as follows.

Given a group G, L is the GRS recommended list of size k, and, for each $i \in G$, $R_{i,test}$ is the set of his/her rated movies in the *test set*, we have that:

- $precision_i@k = \frac{|R_{i,test} \cap L|}{k}$
- $recall_i@k = \frac{|R_{i,test} \cap L|}{|R_{i,test}|}$
- $F1 - score_i = \sum_{k \in \{5,10,20\}} (2 \cdot \frac{precision_i@k \cdot recall_i@k}{precision_i@k + recall_i@k})/3$

---

[3] The *ACM Conference on Recommendation Systems (RecSys)* is the most important international conference in the field of recommendation systems. For more info visit: https://recsys.acm.org/, https://recsys.acm.org/recsys14/challenge-workshop/.

Table 2. Average values for different values of δ and with random values for λ.

	$\delta = 0$	$\delta = 0.25$	$\delta = 0.5$	$\delta = 0.75$	$\delta = 1$
F1 score	0.037 ± 0.044	0.037 ± 0.044	0.037 ± 0.044	0.037 ± 0.044	0.036 ± 0.044
nDCG	0.576 ± 0.216	0.576 ± 0.216	0.576 ± 0.216	0.576 ± 0.217	0.576 ± 0.217

- $nDCG_i = \sum_{k\in\{5,10,20\}}(\sum_{j=1}^{k} \frac{g_{ij}}{max\{1, log_2 j\}})/3$
  where $g_{ij} = \begin{cases} x_{ij} & \text{if } j \in R_{i,test} \\ 0 & \text{otherwise} \end{cases}$

Then, we can define the two measures for the whole group:

$$F1 - score_G = \frac{1}{|G|}\sum_{i\in G} F1 - score_i \tag{9}$$

$$nDCG_G = \frac{1}{|G|}\sum_{i\in G} nDCG_i \tag{10}$$

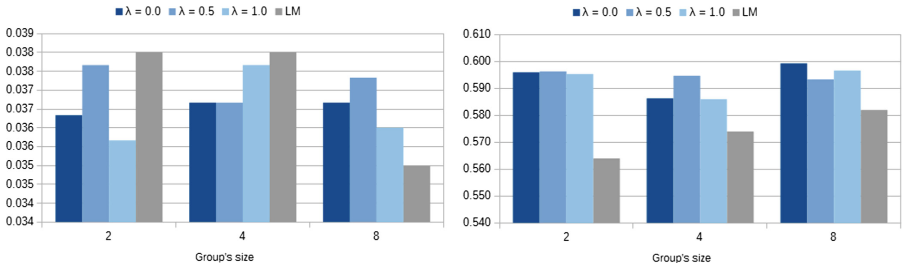
The first analysis is conducted by assigning different random λ values for each user in the generated groups. We reported the average values for the *F1-score* and the *nDCG* in Table 2 averaging on the different group sizes. We can observe that the values are the same independently from the values of the δ parameter. Also grouping with respect to the group sizes, we have not observed any relevant difference. This could due to the random assignment of the λ parameter that has a stronger impact on the results with respect to δ. For this reason, for the second offline study as well as for the pilot study, we set δ equal to 0.5 for giving the same importance to both goals (e.g., social welfare maximization and inequity aversion).

Moreover, we conducted a second analysis by fixing the same value for λ and δ within the groups. Hence, for each experiment, λ can assume a value in $\{0, 0.5, 1\}$. On the same group, we also evaluate the results obtained by applying the Least Misery (LM) aggregation strategy. In detail, given the $x_{i,j}$ individual payoffs (evaluated by the RS), the LM group score $x_{G,j}$, for the item j is computed as follow:

$$x_{G,j} = \min_{i\in G}\{x_{i,j}\} \tag{11}$$

Then the top k elements are selected.

The first evidence of the second experiment is that there is a different, but small, behavior of *F1-score* and the *nDCG* while changing the group size and the λ value, also with respect to LM. By comparing the obtained results for the different values of λ, we were not able to identify a configuration that outperforms the others. For example, in Fig. 2, the results aggregated for different @k values in the case of three different group's sizes are shown. Remember that the results are obtained by averaging the group results with respect to the individual ones. Hence, a user study is necessary.



(a) F1-score (b) nDCG

Fig. 2. *F1-score* and *nDCG* average values for different group's sizes.

6 Pilot Study

To conduct our experiments, we developed a client-server application. Client was an Android[4] app and the server was developed in Java[5] using Spring framework[6] hosted by Tomcat Servlet Engine[7].

6.1 Android Application

An Android application was developed in order to gather the information needed by the server to provide recommendations to single users. In order to simplify the operations, the experiment consists of some sequential steps so that each phase unlocks the next one.

The first duty for the user, when he/she accesses the application, is to sign up/sign into the system. The user signs up to the system by entering a username, password, gender, age, and education level. When the interaction starts, users have first to provide a certain number of movie ratings (at least for 20 movies), with a value in the range $[1, 10]$ (see Fig. 3(a) and (b)) in order to define their profile. The user is provided with an interface to get movie lists and to store movies ratings. If a user is in the training stage, he/she can browse movies by ordering them by most rated or best rated, or by searching for a specific movie (filtering by genre or title). Once the user rated twenty movies, the app automatically shows the personality questionnaire (see Fig. 3(c)).

After this first stage, a user can get movie recommendations from the server. When the server gets the recommendation request, once calculated the best movies for the user, it retrieves additional details about the film, like, for example, the director, writers, actors and genres using OMDb[8] web service. Fortunately, MovieTweetings dataset stores, for each movie, its IMDb id, which can be

---

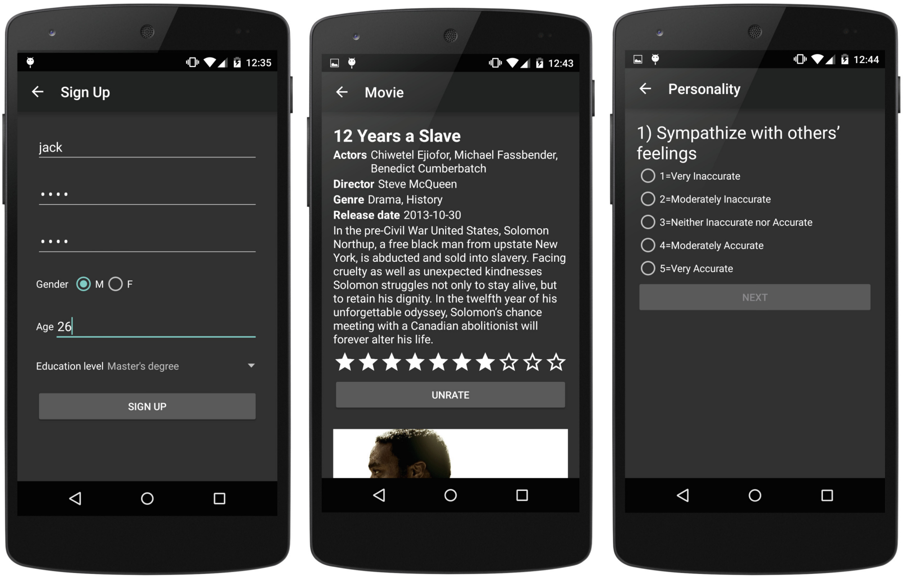
[4] https://www.android.com.

[5] https://www.java.com/.

[6] https://spring.io.

[7] http://tomcat.apache.org.

[8] http://www.omdbapi.com - The Open Movie Database is a free web service to obtain movie information.



jack
••••
••••
Gender ● M ○ F
Age 26
Education level Master's degree

12 Years a Slave
Actors Chiwetel Ejiofor, Michael Fassbender, Benedict Cumberbatch
Director Steve McQueen
Genre Drama, History
Release date 2013-10-30
In the pre-Civil War United States, Solomon Northup, a free black man from upstate New York, is abducted and sold into slavery. Facing cruelty as well as unexpected kindnesses Solomon struggles not only to stay alive, but to retain his dignity. In the twelfth year of his unforgettable odyssey, Solomon's chance meeting with a Canadian abolitionist will forever alter his life.
★★★★★★★☆☆☆

1) Sympathize with others' feelings
○ 1=Very Inaccurate
○ 2=Moderately Inaccurate
○ 3=Neither Inaccurate nor Accurate
○ 4=Moderately Accurate
○ 5=Very Accurate

(a) Sign up (b) Movie rating (c) Personality test

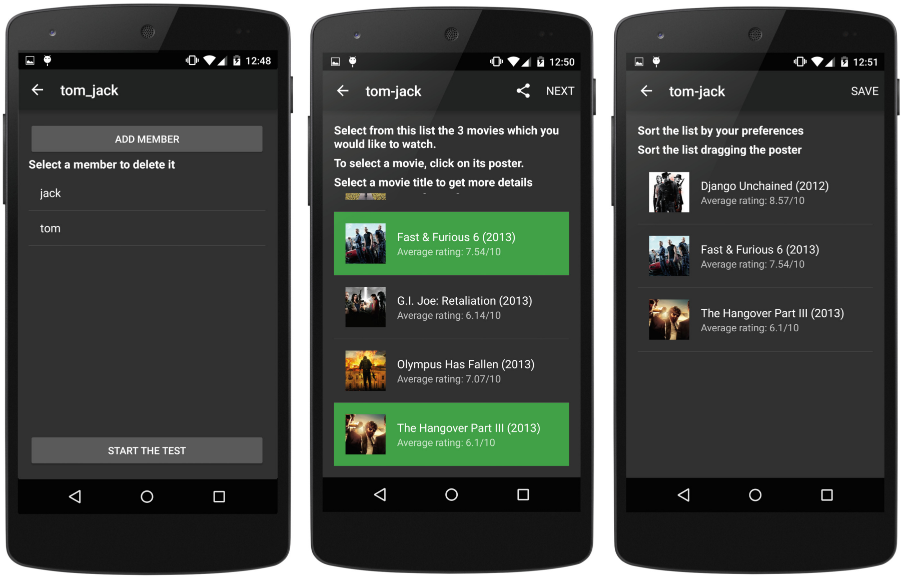
Fig. 3. Application interface for user profiling.

used to address the OMDb service. The Android application shows on the screen the recommendations for the user through textual and graphical descriptions.

6.2 Methodology

The design of this study is a within-subjects, counterbalanced, repeated measures experiment. The goal of our study is to compare the proposed technique, described in the previous paragraphs, with respect to the Least Misery. We selected the latter because it achieves good performance especially for small groups [34] and it deals with inequity adversity.

Once the questionnaire is completed, a user can start the test by completing the following steps. Firstly, a user creates a group giving it a name and adds in it one or more members using their usernames (see Fig. 4(a)). The system, then, recommends a list of 10 movies (see Fig. 4(b)). In order to generate the list of 10 items L for the group, for each technique, the GRS recommends 10 movies ordered in terms of their ratings (the used utility function L_U and least misery L_{LM}). To merge them into one list of ten items, we developed an iterative algorithm that, at each step, adds the first item from each list (respectively, j_{LM} and j_U) (starting from the items with the highest rating) in the output set. In order to avoid duplicates, once an item is inserted in L, it is removed from both L_{LM} and L_U (if it is present).



Select a member to delete it

jack

tom

Select from this list the 3 movies which you would like to watch.

To select a movie, click on its poster.

Select a movie title to get more details

Fast & Furious 6 (2013)
Average rating: 7.54/10

G.I. Joe: Retaliation (2013)
Average rating: 6.14/10

Olympus Has Fallen (2013)
Average rating: 7.07/10

The Hangover Part III (2013)
Average rating: 6.1/10

Sort the list by your preferences

Sort the list dragging the poster

Django Unchained (2012)
Average rating: 8.57/10

Fast & Furious 6 (2013)
Average rating: 7.54/10

The Hangover Part III (2013)
Average rating: 6.1/10

(a) Group creation (b) Recommended movies (c) Sorting page

Fig. 4. Group recommendation phases.

From this list, the group has to collectively choose three movies that they would like to see together (see Fig. 4(b)). Finally, the group has to sort the three selected movies (see Fig. 4(c)) in their joint preference order.

6.3 Evaluation Metrics

Since each group jointly selects its favorite 3 movies out of a 10 movies set, in order to evaluate and compare our method with LM, we considered the following metrics.

precision@3 is the ratio between the number of movies guessed by the GRS (using a specific method) and the sum of the latter and the remaining movies (only considering the first three movies). If G is the set of the groups that participated in the experiment, I_g and P_g represent, respectively, the 3 movies selected by group g and the 3 movies with the highest prediction, then:

$$precision@3 = \frac{1}{|G|} \sum_{g \in G} \frac{|I_g \cap P_g|}{|I_g \cap P_g| + (3 - |I_g \cap P_g|)} \tag{12}$$

nDCG@3 evaluates the ranking of predicted movies with respect to the real ranking j chosen by groups.

$$nDCG@3 = \frac{1}{|G|}\sum_{g\in G}\sum_{j=1}^{3}\frac{rel_{gi_j}}{\max(1, log_2 j)} \tag{13}$$

where,

$$rel_{gi_j} = \begin{cases} 1 & \text{if } i_j \in I_g \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

x_success@3, for each group, is 1 if the algorithm guessed at least x movies in the 3 selected by the group. With $1 \leq x \leq 3$:

$$x\_success@3 = \frac{1}{|G|}\sum_{g\in G} x\_success@3_g \tag{15}$$

where,

$$x\_success@3_g = \begin{cases} 1 & \text{if } |I_g \cap P_g| \geq x \\ 0 & \text{otherewise} \end{cases} \tag{16}$$

6.4 Result Analysis

Experiments lasted about two weeks and, as summarized in Table 3, we recruited 68 users (48 groups) with an average age of 27 years, the most of whom were students.

Table 3. User stats.

Number of users	68
Average age	26.9
Minimum age	14
Maximum age	55
Males	45
Females	23
Middle school	1
High school	9
Bachelor students - undergraduate	26
Bachelor students - graduate	11
Master students - undergraduate	11
Master students - graduates	10

Table 4. Group stats.

# members	Amount
2	22
3	23
4	2
5	1
Total	48

Group members were directly selected by one of the users, as described above, and their intersection is not necessarily empty (e.g., some users joined more than one group). A single group was allowed to join the test only for one time. The number of considered group was 48 with an average number of members equals to 2.6. In Table 4, we reported the number of groups considered in the experiments for each group dimension.

First of all, we evaluated the average value of the obtained λ for each participant in the experiments. Such value is 0.7, hence, in average, questionnaire results show that the selected users obtained a high value of agreeableness (no one obtained a α score smaller that 2.75).

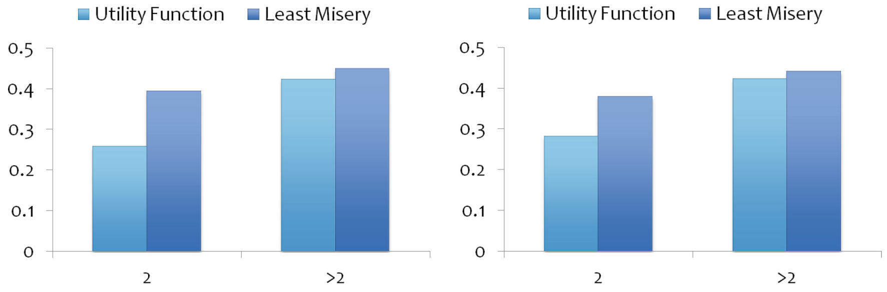
Moreover, when comparing the two techniques, we analyzed their results separately for the groups of dimension two and for the groups of more than two members. The main reason for this choice is that about the half of the groups were composed of two members (see Table 4) and it is known that LM shows the best performances in this case.

precision@3. Results of precision@3 are summarized in Fig. 5(a). From the charts, we can see the better performance of LM on two members' groups. It is not a surprise since LM excels in cases like this. ANOVA test confirms that difference between the two techniques is significant ($F = 3.076$, p-value = 0.09).

Regarding groups with more than two members, once again LM is better than our technique, but difference, in this case, is too small to be significant for ANOVA test ($F = 0.11$, p-value = 0.74).

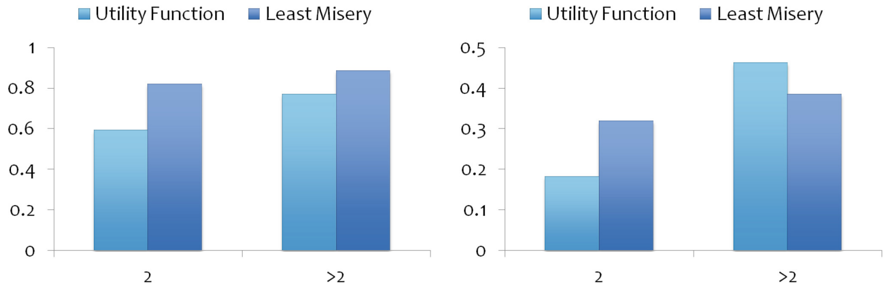
nDCG@3. As for the previous metrics, LM overcomes the proposed technique (see Fig. 5(b)), but not enough according ANOVA ($F = 1.547$, p-value = 0.22 for the two members' groups and $F = 0.056$, p-value = 0.81 for the others).

1_success@3. We recall that 1_success@3 counts the number of times that the algorithm guessed at least one movie among the three selected by the group. LM is always better than our technique (see Fig. 6(a)), but these results provide significant differences only when considering all the 48 groups together ($F = 3.847$, p-value = 0.05).



(a) precision@3 (b) ndcg@3

Fig. 5. Precision and nDCG for the pilot study.



(a) 1_success@3 (b) 2_success@3

Fig. 6. 1_success@3 and 2_success@3 for the pilot study.

2_success@3. In this case, our method was more accurate in groups with more than two members (see Fig. 6(b)). Unfortunately ANOVA shows that these differences are due to chance ($F = 0.305$, p-value $= 0.58$).

3_success@3. It is really rare that a technique is able to guess exactly the three movies chosen by the group. However, in some cases, it happened. Once a time LM wins but with no significant differences.

Finally, we evaluated the Pearson correlations on result distributions that show that the two techniques are not linearly dependent because, in most cases, its value is near zero. This result means that, in some circumstances, the proposed utility function shows a better performance than LM and in other cases the opposite occurs. Therefore, in certain groups, users do care to minimize the group unsatisfaction (i.e., the LM goal) regardless of the *agreeableness* value.

7 Conclusions and Future Work

Group recommendation algorithms have to take into account the members' relationships as well as other characteristics that may influence the users' decision making while being in a group. In this work, starting from a model of other-regarding preferences, we introduced a new method to predict which items are suitable for groups of users, taking into account users' personality. In particular, we evaluated the role of the *agreeableness* factor (e.g., one of the traits of the Five-Factor Model), in order to weigh the importance of user's gain with respect to the global satisfaction.

Groups recommendation systems lack the appropriate dataset to be evaluated upon. On the contrary, user studies have the disadvantages of being expensive both in term of recruiting the proper users and engaging them, when they are volunteers. In this sense, founding the appropriate number of groups, with varying dimensions, and that is a good representative of the considered population is a challenging process.

Hence, to evaluate this approach, we firstly conducted an offline evaluation process. From this analysis, it emerged that, with a random distribution of the agreeableness trait, the proper balancing among the social welfare and the least misery components of the considered utility function may not have a relevant weight in the determination of the items to suggest to the group. On the contrary, the variation of the trait value in the population produces different results (also with respect to the least misery), but without a clear trend. Moreover, the used dataset contains only individual ratings that do not consider group choices.

We also conducted a pilot user study on movie recommendations, where we compared the results of the proposed approach with respect to LM. As expected, for two users groups, LM is the best choice. In the other cases, we cannot say the same thing since the two methods are comparable and show a similar performance. We foresee that our utility function will improve its effectiveness proportionally to the group size: the larger is the group, the greater will count altruism in the final decision. Hence, in future works, although our work focuses on small groups dynamics, we should try to encourage users to create larger groups in order to better support our hypothesis.

Moreover, we think that these first results could be affected by the experiments methodology. As already said, the 10 movies recommended to the groups are obtained by a sort of union of two lists independently generated by the two different techniques. Thus, in this set, there are movies chosen by both LM and the proposed utility function. Maybe, a between-subject experiment (by assigning the result of a single technique to each group) would lead us to different results. Moreover, the low number of experiments had a relevant impact in the significance analysis of the results. Finally, since most of the groups were composed of two or three members, we foresee that, with larger groups, our technique could have obtained better results.

Finally, as future work, we could study how to use other personality factors to build another utility function. We saw that *extraversion* is correlated to the leadership of a group, so in a decision process, it could be very critical. Furthermore, even *openness* could be decisive in such cases, because it could have the same weight of the *agreeableness* in our function. Open people are glad to try new experiences, so they could agree to view a movie for which the recommender system predict a low value for them, but the opposite for other members.

References

1. Anderson, C., John, O.P., Keltner, D., Kring, A.M.: Who attains social status? effects of personality and physical attractiveness in social groups. J. Pers. Soc. Psychol. **81**(1), 116 (2001)
2. Baltrunas, L., Makcinskas, T., Ricci, F.: Group recommendations with rank aggregation and collaborative filtering. In: Proceedings of the ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, 26–30 September, pp. 119–126 (2010)
3. Barile, F., Caso, A., Rossi, S.: Group recommendation for smart applications: a multi-agent view of the problem. In: XV Workshop Dagli Oggetti agli Agenti, vol. 1260. CEUR-WS.org (2014)
4. Berkovsky, S., Freyne, J.: Group-based recipe recommendations: analysis of data aggregation strategies. In: RecSys 2010 - Proceedings of the 4th ACM Conference on Recommender Systems, pp. 111–118. CSIRO Tasmanian ICT Center (2010)
5. Berry, S., Fazzio, S., Zhou, Y., Scott, B., Francisco-Revilla, L.: Netflix recommendations for groups. Proc. Am. Soc. Inf. Sci. Technol. **47**(1), 1–3 (2010)
6. Brody, L.R.: The socialization of gender differences in emotional expression: display rules, infant temperament, and differentiation. In: Gender and Emotion: Social Psychological Perspectives, pp. 24–47 (2000)
7. Burke, R.: Hybrid recommender systems: survey and experiments. User Model. User-Adap. Inter. **12**(4), 331–370 (2002)
8. Cantador, I., Castells, P.: Group recommender systems: new perspectives in the social web. In: Recommender Systems for the Social Web. ISRL, vol. 32, pp. 139–157. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25694-3_7
9. Charness, G., Rabin, M.: Understanding social preferences with simple tests. Q. J. Econ. **117**(3), 817–869 (2002)
10. Costa, P.T., MacCrae, R.R.: Revised NEO Personality Inventory (NEO PI-R) and NEO Five-Factor Inventory (NEO FFI): Professional Manual. Psychological Assessment Resources, Odessa (1992)
11. Costa, P.T., McCrae, R.R.: Primary traits of Eysenck's pen system: three-and five-factor solutions. J. Pers. Soc. Psychol. **69**(2), 308 (1995)
12. Cox, J.C., Friedman, D., Gjerstad, S.: A tractable model of reciprocity and fairness. Games Econ. Behav. **59**(1), 17–45 (2007)
13. Darley, W.K., Blankson, C., Luethge, D.J.: Toward an integrated framework for online consumer behavior and decision making process: a review. Psychol. Market. **27**(2), 94–116 (2010)
14. Donnellan, M.B., Oswald, F.L., Baird, B.M., Lucas, R.E.: The mini-IPIP scales: tiny-yet-effective measures of the big five factors of personality. Psychol. Assess. **18**(2), 192 (2006)
15. Dooms, S., De Pessemier, T., Martens, L.: Movietweetings: a movie rating dataset collected from twitter. In: Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013 (2013)
16. Dunn, G., Wiersema, J., Ham, J., Aroyo, L.: Evaluating interface variants on personality acquisition for recommender systems. In: Houben, G.-J., McCalla, G., Pianesi, F., Zancanaro, M. (eds.) UMAP 2009. LNCS, vol. 5535, pp. 259–270. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02247-0_25
17. Elfenbein, H.A.: The many faces of emotional contagion an affective process theory of affective linkage. Organ. Psychol. Rev. **4**, 326–362 (2014). https://doi.org/10.1177/2041386614542889

18. Fehr, E., Schmidt, K.: The economics of fairness, reciprocity and altruism - experimental evidence and new theories. In: Kolm, S., Ythier, J.M. (eds.) Handbook of the Economics of Giving, Altruism and Reciprocity, vol. 1, chap. 08, 1st edn., pp. 615–691. Elsevier (2006)
19. Gartrell, M., Xing, X., Lv, Q., Beach, A., Han, R., Mishra, S., Seada, K.: Enhancing group recommendation by incorporating social relationship interactions. In: Proceedings of the 16th ACM International Conference on Supporting Group Work, pp. 97–106. ACM (2010)
20. Goldberg, L.R.: The development of markers for the big-five factor structure. Psychol. Assess. **4**(1), 26 (1992)
21. Gosling, S.D., Rentfrow, P.J., Swann, W.B.: A very brief measure of the big-five personality domains. J. Res. Pers. **37**(6), 504–528 (2003)
22. Hu, R., Pu, P.: Enhancing collaborative filtering systems with personality information. In: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys 2011, pp. 197–204. ACM (2011)
23. Jameson, A., Smyth, B.: Recommendation to groups. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. LNCS, vol. 4321, pp. 596–627. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72079-9_20
24. Kilmann, R.H., Thomas, K.W.: Developing a forced-choice measure of conflict-handling behavior: the “mode” instrument. Educ. Psychol. Measur. **37**(2), 309–325 (1977)
25. Kompan, M., Bielikova, M.: Social structure and personality enhanced group recommendation. In: Proceedings of the 2nd Workshop Emotions and Personality in Personalized Services (EMPIRE). CEUR-WS.org (2014)
26. Levine, J.M., Moreland, R.L.: Small Groups: Key Readings. Psychology Press, New York (2008)
27. Ma, Z.: Exploring the relationships between the big five personality factors, conflict styles, and bargaining behaviors. In: IACM 18th Annual Conference (2005)
28. Masthoff, J.: Group recommender systems: combining individual models. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 677–702. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-85820-3_21
29. Masthoff, J., Gatt, A.: In pursuit of satisfaction and the prevention of embarrassment: affective state in group recommender systems. User Model. User-Adap. Inter. **16**(3–4), 281–319 (2006)
30. McCrae, R.R., Costa, P.T.: Updating norman’s “adequacy taxonomy”: intelligence and personality dimensions in natural language and in questionnaires. J. Pers. Soc. Psychol. **49**(3), 710 (1985)
31. McCrae, R.R., Costa, P.T.: Validation of the five-factor model of personality across instruments and observers. J. Pers. Soc. Psychol. **52**(1), 81 (1987)
32. McCrae, R.R., Costa, P.T.: The structure of interpersonal traits: Wiggins’s circumplex and the five-factor model. J. Pers. Soc. Psychol. **56**(4), 586 (1989)
33. Nunes, M.A.S., Hu, R.: Personality-based recommender systems: an overview. In: Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys 2012, pp. 5–6. ACM, New York (2012)
34. O’Connor, M., Cosley, D., Konstan, J.A., Riedl, J.: Polylens: a recommender system for groups of users. In: Proceedings of the 7th European Conference on CSCW, pp. 199–218 (2001)

35. Quijano-Sánchez, L., Recio-Garcıa, J.A., Dıaz-Agudo, B.: Group recommendation methods for social network environments. In: 3rd Workshop on Recommender Systems and the Social Web 5th ACM International Conference on Recommender Systems, RecSys, vol. 11 (2011)
36. Quijano-Sanchez, L., Recio-Garcia, J.A., Diaz-Agudo, B., Jimenez-Diaz, G.: Social factors in group recommender systems. ACM Trans. Intell. Syst. Technol. **4**(1), 1–30 (2013)
37. Recio-Garcia, J.A., Jimenez-Diaz, G., Sanchez-Ruiz, A.A., Diaz-Agudo, B.: Personality aware recommendations to groups. In: Proceedings of the Third ACM Conference on Recommender Systems, RecSys 2009, pp. 325–328. ACM, New York (2009)
38. Rentfrow, P.J., Goldberg, L.R., Zilca, R.: Listening, watching, and reading: the structure and correlates of entertainment preferences. J. Pers. **79**(2), 223–258 (2011)
39. Rossi, S., Barile, F., Caso, A., Rossi, A.: Pre-trip ratings and social networks user behaviors for recommendations in touristic web portals. In: Monfort, V., Krempels, K.-H., Majchrzak, T.A., Turk, Ž. (eds.) WEBIST 2015. LNBIP, vol. 246, pp. 297–317. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30996-5_15
40. Rossi, S., Caso, A., Barile, F.: Combining users and items rankings for group decision support. In: Bajo, J., et al. (eds.) Trends in Practical Applications of Agents, Multi-Agent Systems and Sustainability. AISC, vol. 372, pp. 151–158. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19629-9_17
41. Rossi, S., Cervone, F.: Social utilities and personality traits for group recommendation: a pilot user study. In: Proceedings of the 8th International Conference on Agents and Artificial Intelligence, pp. 38–46 (2016)
42. Rossi, S., Di Napoli, C., Barile, F., Liguori, L.: Conflict resolution profiles and agent negotiation for group recommendations. In: XVII Workshop Dagli Oggetti agli Agenti, vol. 1664. CEUR-WS.org (2016)
43. Sager, K.L., Gastil, J.: The origins and consequences of consensus decision making: a test of the social consensus model. South. Commun. J. **71**(1), 1–24 (2006)
44. Salehi-Abari, A., Boutilier, C.: Empathetic social choice on social networks. In: 13th International Conference on Autonomous Agents and Multiagent Systems, pp. 693–700 (2014)
45. Scheibehenne, B., Greifeneder, R., Todd, P.M.: Can there ever be too many options? A meta-analytic review of choice overload. J. Consum. Res. **37**(3), 409–425 (2010)
46. Senot, C., Kostadinov, D., Bouzid, M., Picault, J., Aghasaryan, A., Bernier, C.: Analysis of strategies for building group profiles. In: De Bra, P., Kobsa, A., Chin, D. (eds.) UMAP 2010. LNCS, vol. 6075, pp. 40–51. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13470-8_6
47. Tkalcic, M., Chen, L.: Personality and recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 715–739. Springer, Boston (2015). https://doi.org/10.1007/978-1-4899-7637-6_21
48. Wang, J., Liu, Z., Zhao, H.: Group recommendation based on the pagerank. J. Netw. **7**(12), 2019–2024 (2012)
49. Wu, W., Chen, L., He, L.: Using personality to adjust diversity in recommender systems. In: Proceedings of the 24th ACM Conference on Hypertext and Social Media, HT 2013, pp. 225–229. ACM, New York (2013)

Two-Stage Reinforcement Learning Algorithm for Quick Cooperation in Repeated Games

Wataru Fujita[1](✉), Koichi Moriyama[2], Ken-ichi Fukui[1,3],
and Masayuki Numao[1,3]

[1] Graduate School of Information Science and Technology,
Osaka University, 1-5, Yamadaoka, Suita, Osaka 565-0871, Japan
{fujita,fukui,numao}@ai.sanken.osaka-u.ac.jp
[2] Graduate School of Engineering, Nagoya Institute of Technology, Gokiso-cho,
Showa-ku, Nagoya 466-8555, Japan
moriyama.koichi@nitech.ac.jp
[3] The Institute of Scientific and Industrial Research, Osaka University,
8-1, Mihogaoka, Ibaraki, Osaka 567-0047, Japan

Abstract. People often learn their behavior from its outcome, e.g., success and failure. Also, in the real world, people are not alone and many interactions occur among people every day. To model such learning and interactions, let us consider reinforcement learning agents playing games. Many researchers have studied a lot of reinforcement learning algorithms to obtain good strategies in games. However, most of the algorithms are "suspicious", i.e., focusing on how to escape from being exploited by greedy opponents. Therefore, it takes long time to establish cooperation among such agents. On the other hand, if the agents are "innocent", i.e., prone to trust others, they establish cooperation easily but are exploited by acquisitive opponents. In this work, we propose an algorithm that uses two complementary, "innocent" and "suspicious" algorithms in the early and the late stage, respectively. The algorithm allows the agent to cooperate with good associates quickly as well as treat greedy opponents well. The experiments in ten games showed us that the proposed algorithm successfully learned good strategies quickly in nine games.

Keywords: Multi-agent · Reinforcement learning · Game theory

1 Introduction

Humans make a variety of decisions in their daily lives. In social situations in which a person's decision depends on others, there are complicated mutual relations such as competition and cooperation among people. Many researchers have widely studied game theory that models relations as "games" and analyzed rational decision-making in games.

© Springer International Publishing AG, part of Springer Nature 2018
N. T. Nguyen et al. (Eds.): TCCI XXVIII, LNCS 10780, pp. 48–65, 2018.
https://doi.org/10.1007/978-3-319-78301-7_3

Furthermore, humans have an instinctive desire for survival; therefore, learning through trial-and-error to avoid harmful and unpleasant states and approach beneficial and pleasant states. Such learning occurs given a learning mechanism in the human brain. Many researchers study reinforcement learning algorithms that model this learning mechanism.

Let us consider people who learn their behavior from the result of interacting with others. As a model of this, we discuss reinforcement learning agents that play games. A lot of (multi-agent) reinforcement learning algorithms that perform well in various games have already been proposed; however, such algorithms typically require a large number of interactions to learn appropriate behavior in games. People in the real world must quickly learn to make reasonable decisions, because the world changes rapidly. Existing algorithms have various features, each with its advantages and disadvantages. These algorithms can be complementary. Therefore, in this study, we construct an algorithm that learns quickly and performs well in any type of game by combining features from multiple algorithms.

Aside from this introductory section, the structure of this paper is as follows. In Sect. 2, we introduce games and learning algorithms used in later sections. In Sect. 3, we construct a new learning algorithm that learns quickly and performs well in various games by combining two learning algorithms. Next, we evaluate our proposed algorithm by conducting experiments, as described in Sect. 4. In Sect. 5, we discuss our proposed algorithm. We introduce related works in Sect. 6 to show the relative position of our algorithm. Finally, in Sect. 7, we conclude our paper and provide avenues for future work.

2 Background

In this section, we introduce game theory that models interactions among people and reinforcement learning algorithms that model trial-and-error learning for adaptation to a given environment.

2.1 Game Theory

We, as humans, are always making decisions as to what to do next to achieve our desired purpose or goals. In social environments, every decision is affected by the decisions of other people. Game theory mathematically analyzes the relationship among such decisions.

A game in game theory consists of the following four elements [7]:

1. Rules that govern the game;
2. Players who decide what to do;
3. Action strategies of the players; and
4. Payoffs given to the players as a result of their decisions.

Game theory analyzes how players behave in an environment in which their actions mutually influence one another. We focus on two-person simultaneous games in this study.

Table 1. An example of the prisoner's dilemma game

	Cooperation	Defection
Cooperation	0.6, 0.6	0.0, 1.0
Defection	1.0, 0.0	0.2, 0.2

In a two-person simultaneous game, two players simultaneously choose actions based on their given strategies. After both players choose their respective actions, each player is given a payoff determined by the "joint action" of both players. Since the payoff of each player is determined by not only his or her action but also the other player's action, it is necessary to deliberate the other player's action to maximize payoffs.

Note that all games used in this research are non-cooperative, and players choose their individual strategies based on their own payoffs and all players' past actions. More specifically, a player chooses his or her own action based on his or her payoff and the action of the other players in the previous period.

A Nash equilibrium is defined as a combination of actions in which no player is motivated to change his or her strategy. Let us consider the "prisoner's dilemma" game summarized in Table 1. Here, the row and column correspond to the actions of Players 1 and 2, respectively; they gain the left and right payoffs, respectively, corresponding to the joint action in the matrix.

According to the payoff matrix, the player should choose the Defection action regardless of the other player's action, because it always yields higher payoffs than the Cooperation action. Since the other player considers the same, both players choose Defection, and finally, the combination of actions (i.e., Defection, Defection) becomes a Nash equilibrium.

Conversely, if both players select Cooperation, both payoffs can be raised to 0.6 from 0.2; however, it is very difficult for both players to choose Cooperation, because the combination (i.e., Cooperation, Cooperation) is not an equilibrium and each player is motivated to choose Defection. Moreover, even if a player overcomes this motive for a certain reason, he or she will yield a payoff of zero if the partner chooses Defection. The prisoner's dilemma game shows that the individual's rationality differs from that of social rationality in a social situation.

2.2 Reinforcement Learning

Reinforcement learning [11] is a learning method that learns strategies through interacting with the given environment. An agent is defined as a decision-making entity, while the environment is everything external to the agent that interacts with the agent. The agent interacts with the environment at discrete time steps, i.e., $t = 0, 1, 2, 3, ...$. At each time step t, the agent recognizes the current state $s^t \in S$ of the environment, where S is a set of possible states, and decides an action $a^t \in A(s^t)$ based on the current state, where $A(s^t)$ is a set of actions selectable in a state s^t. Then, the agent takes the action. It gives the agent a reward $r^{t+1} \in \Re$ and changes the state to a new state s^{t+1}.

The probability that the agent chooses a possible action a in a state s is shown as a strategy $\pi^t(s, a)$. Reinforcement learning algorithms update the strategy π^t, or *action values* that indirectly change the strategy, at each time step.

One of the most popular reinforcement learning algorithm is Q-learning [12]. It learns an action value function $Q(s, a)$ (called Q-values) by using the following rule.

$$Q^{t+1}(s^t, a^t) = Q^t(s^t, a^t) + \alpha[r^{t+1} + \gamma V^t(s^{t+1}) - Q^t(s^t, a^t)], \tag{1}$$

$$V^t(s^{t+1}) = \max_{a \in A(s^{t+1})} Q^t(s^{t+1}, a), \tag{2}$$

where $\alpha \in (0, 1]$ is a parameter called the learning rate. Under some conditions, the Q-values stochastically converges to the expected sum of rewards, discounted by $\gamma \in [0, 1)$, derived from the optimal strategy [12].

At every time step, the agent derives its strategy from the Q-values. One of the most popular method is called ϵ-greedy, which chooses an action with the maximum Q-value with probability $1 - \epsilon$, or a random action with probability ϵ.

2.3 Three Foundational Learning Algorithms

Here we introduce three learning algorithms that form the basis for our proposal.

M-Qubed: M-Qubed [3] is an excellent state-of-the-art reinforcement learning algorithm that can learn to cooperate with associates (i.e., other players) and avoid being exploited unilaterally in various games. M-Qubed updates the Q-value with Sarsa [8]. Sarsa uses the same update rule of Q-learning (Eq. 1), but the function V is different:

$$V^t(s^{t+1}) = \sum_{a \in A(s^{t+1})} \pi^t(s^{t+1}, a) Q^t(s^{t+1}, a). \tag{3}$$

M-Qubed assumes that the state s is defined as the latest joint action of the agent and its associates, and the reward r is in $[0, 1]$.

M-Qubed consists of the following three components. Note that the *maximin payoff* is a secured payoff regardless of the associates, given by the *maximin strategy* that maximizes the minimum payoff based on the payoff definition of the game.

Profit pursuit

This gives a pure strategy that chooses an action with the maximum Q-value if the Q-value is larger than the discounted sum of the maximin payoff; otherwise, it gives the maximin strategy.

Loss aversion

This gives a pure strategy that chooses an action with the maximum Q-value if the accumulated loss is less than a given threshold; otherwise, it gives the maximin strategy. The accumulated loss is the difference between the accumulated payoff and the accumulated maximin payoff. The threshold is set in proportion to the number of possible states and joint actions.

Optimistic search

The strategy of "Profit pursuit" can acquire high payoffs for the moment; however, it tends to produce myopic actions. The strategy of "Loss aversion" cannot lead cooperation with associates, which may yield higher payoffs. To solve these problems, M-Qubed sets the initial Q-values to their highest possible discounted reward $1/(1-\gamma)$, thereby learning wider strategies.

First of all, M-Qubed initializes the Q-values with the "Optimistic search". After every update of Q-values, M-Qubed calculates a weighted average of the two strategies given by "Profit pursuit" and "Loss aversion".

In addition, if all recently visited states have low Q-values, the strategies of the agent and its associates may remain at a local optimum. Then, the agent must explore further to find a solution that may give a higher payoff. Hence, in this case, M-Qubed gives a strategy that is a weighted average of the above strategy and a completely random strategy.

Satisficing Algorithm: Satisficing algorithm (S-alg) [10] is an algorithm that calculates a value called the *aspiration level* of the agent. The agent continues to take an action that gives payoffs more than its aspiration level. The algorithm is shown in Algorithm 1. S-alg is an algorithm that enables an agent to learn to take cooperative actions with its associates.

Algorithm 1. Satisficing algorithm

```
Variables: $t$: round, $l^t$: aspiration level at $t$, $a^t$: action at $t$, $r^t$: payoff at $t$,
Parameters: $A$: set of available actions, $\lambda \in (0,1)$: learning rate,
  $R_{max}$: maximum available payoff (or sufficiently large real number)

  $t \leftarrow 1$
  $l^1 \leftarrow Random \in [R_{max}, 2R_{max}]$
  $a^1 \leftarrow Random \in A$
  repeat
    Take $a^t$ and receive $r^t$
    if $r^t \geq l^t$ then
      $a^{t+1} \leftarrow a^t$
    else
      $a^{t+1} \leftarrow Random \in A$
    end if
    $l^{t+1} \leftarrow \lambda l^t + (1-\lambda) r^t$
    $t \leftarrow t + 1$
  until Finish
```

BM Algorithm: BM algorithm [4] is a reinforcement learning algorithm that maximizes payoffs by combining M-Qubed and S-alg. It produces a new strategy that combines the strategies derived from the two internal algorithms. Both algorithms simultaneously update their functions, i.e., the Q-value and the aspiration level.

BM algorithm uses Boltzmann multiplication [13] as the method of combination. Boltzmann multiplication multiplies strategies of all internal algorithms for each available action and determines the ensemble strategy by the Boltzmann distribution. The preference value of an action in a state, p^t, is defined as

$$p^t(s^t, a) \equiv \prod_j \pi_j^t(s^t, a) \tag{4}$$

and the resulting ensemble strategy is defined as

$$\pi^t(s^t, a) = \frac{p^t(s^t, a)^{\frac{1}{\tau}}}{\sum_{x \in A(s^t)} p^t(s^t, x)^{\frac{1}{\tau}}}, \tag{5}$$

where π_j^t is a strategy of an internal algorithm j, $a \in A(s^t)$ is a possible action in the state s^t, and τ is a temperature parameter. After calculating the ensemble strategy, the agent selects an action, and then all internal algorithms learn from the result of this selected action.

Since S-alg yields only pure strategies, all actions except for the chosen one have zero probability. The M-Qubed part becomes meaningless when the Boltzmann multiplication is used to combine M-Qubed and S-alg without consideration. Therefore, the S-alg in BM algorithm yields a mixed strategy in which the chosen action is played with probability 0.99.

3 Proposed Algorithm

In this study, we consider reinforcement learning agents that play games. M-Qubed, BM, and S-alg agents perform well in some games, but have problems in other games, as summarized below:

- M-Qubed requires a long time to learn, because it has multiple strategies and needs to decide which one is used. Therefore, the average payoff becomes less than S-alg in a game having only one suitable solution that is in cooperation with the associates.
- BM algorithm performs better than M-Qubed, but it is still slow in search because the internal S-alg cannot completely compensate for the slowness of M-Qubed.
- If the associates are greedy, S-alg is exploited unilaterally, because it decreases the aspiration level, and then S-alg is satisfied with low payoffs.
- Due to insufficient search, S-alg may be satisfied with the second-best payoff.

These algorithms do not have sufficient performance for a variety of reasons; however, their positive abilities are complementary. BM, which is not exploited due to the "Loss aversion" strategy and explores the given environment thoroughly enough, can compensate for the weakness of S-alg that tends to be exploited. Conversely, S-alg, which quickly learns to cooperate, can compensate for the weakness of BM, i.e., its slowness of search.

The agent should cooperate with a reliable associate quickly but avoid exploitation by a greedy opponent. Therefore, we combine BM and S-alg to construct a two-stage reinforcement learning algorithm that quickly learns good strategies in various games. We call our proposed algorithm *J-algorithm* (J-alg). J-alg has an Exploration stage and a Static stage. The Exploration stage is represented by S' algorithm (S'-alg), which is a slightly modified version of S-alg. Similarly, the Static stage is represented by BM algorithm. In the following subsections, we introduce S'-alg and J-alg.

3.1 S' Algorithm

We focused on S-alg to play a key role in the Exploration stage, because S-alg can learn to cooperate quickly; however, S-alg tends to cover an insufficient search space and be exploited by a myopic strategy. Suppose that the S-alg agents play the Security game (SG) shown in Table 2. If the aspiration level of the row agent is smaller than 0.84, the agent loses its motivation to change its action from x. Consequently, both players are satisfied with a payoff that is not the largest one (1.0, 0.67). S-alg stops searching for other actions and is therefore prone to an insufficient search.

To solve this problem, we slightly modify S-alg; this new algorithm is called the *S' algorithm* (S'-alg). If the agent receives the maximin payoff in the learning phase where the aspiration level is decreasing, S'-alg prohibits the agent from choosing the action again in the next round. The modified algorithm is shown in Algorithm 2. This change stochastically forces the agent to take other actions to escape from a local solution and potentially find a better one.

3.2 Integrating the Two Stages

J-alg uses S'-alg for the Exploration stage and BM for the Static stage. The J-alg agent starts in the Exploration stage. After the joint action converges, the algorithm switches to the Static stage and resets the Q-values to choose the converged action more often. If the joint action does not converge in the first t_c

Table 2. Security game (SG)

	z	w
x	0.84, 0.33	0.84, 0.0
y	0.0, 1.0	1.0, 0.67

Algorithm 2. S' algorithm

Variables: t: round, l^t: aspiration level at t, a^t: action at t, r^t: payoff at t,
Parameters: A: set of available actions, $\lambda \in (0,1)$: learning rate,
R_{max}: maximum available payoff (or sufficiently large real number),
$p \in [0,1]$: probability of prohibition

```
$t \leftarrow 1$
$l^1 \leftarrow Random \in [R_{max}, 2R_{max}]$
$a^1 \leftarrow Random \in A$
repeat
    Take $a^t$ and receive $r^t$
    if $r^t \geq l^t$ then
        $a^{t+1} \leftarrow a^t$
    else if $r^t =$ maximin payoff and $Random \in [0,1) < p$ then
        $a^{t+1} \leftarrow Random \in A \backslash \{a^t\}$
    else
        $a^{t+1} \leftarrow Random \in A$
    end if
    $l^{t+1} \leftarrow \lambda l^t + (1-\lambda) r^t$
    $t \leftarrow t+1$
until Finish
```

Algorithm 3. J-algorithm

Variables: t: round, s: state, a: action, a_c: my action when the joint action converges,
r_c: payoff when the joint action converges, *stage*: current stage
Parameters: $\delta \in [0,1]$: reduction rate

```
$t \leftarrow 1$
$stage \leftarrow$ Exploration
repeat
    if $stage =$ Exploration then
        if $t < t_c$ then
            if the joint action has converged then
                for all $s$ and $a$ do
                    if $a = a_c$ then
                        $Q(s,a) \leftarrow r_c/(1-\gamma)$
                    else
                        $Q(s,a) \leftarrow \delta \times r_c/(1-\gamma)$
                    end if
                end for
                $stage \leftarrow$ Static
            end if
        else
            $stage \leftarrow$ Static
        end if
    end if
    Run a round in the $stage$
    $t \leftarrow t+1$
until Finish
```

rounds, the algorithm simply changes to the Static stage without changing the Q-values. The algorithm is shown in Algorithm 3. Even though the J-alg agent is exploited in the Exploration stage, it will recover in the Static stage via BM algorithm that can evade a loss by using "Loss aversion" strategy.

4 Experiments

To confirm the performance of J-alg, we conducted experiments using 10 two-person two-action matrix games used in the M-Qubed paper [3] shown in Table 3. Here, the agent was able to observe only the previous joint action and its own payoff. We compared J-alg with Q-learning (QL), Q-learning with the optimistic search (QLOP), M-Qubed, S-alg, and BM. QL and QLOP used the ϵ-greedy method to choose actions. We set the initial Q-value of QL randomly and that of QLOP to the highest possible discounted payoff, $1/(1-\gamma)$. The learning rates (α) of QL and QLOP were both set to 0.1 and the exploration rates (ϵ) were both set to 0.1. We also set the M-Qubed parameters to be identical to the original ones. The temperature parameter τ of the BM algorithm was set to 0.2. Further, we set the following parameters of S'-alg and J-alg: the learning rate $\lambda = 0.99$, the prohibition probability $p = 0.3$, the reduction rate $\delta = 0.99$, and the stage-shift threshold time $t_c = 500$. We considered that a certain joint action converged when it had continued 30 rounds. Here, we call the *maximum joint action* a joint action by which the sum of both player's payoffs becomes maximum (which is shown in bold italic typeface in the table).

4.1 Experiment 1: Self-play

Our first experiment was conducted in games where both players are agents with the same algorithm, i.e., in self-play. Agents played one of the 10 games for 30000 rounds, iterating 50 times for each game. We then compared the normalized average payoffs of the six methods. Note that the earlier the actions of agents converge to the maximum joint action, the larger the normalized average payoffs become. Table 4 shows the normalized average payoffs in the games, each of which is the average payoff divided by that of the maximum joint action. If the normalized average payoff is close to one, it shows that the agent quickly learned the maximum joint action in the game.

From the table, we observe that J-alg gained high payoffs and quickly learned the maximum joint action of nine games in self-play. It is a result of good combination of the "innocent" method (S'-alg) and the "suspicious" method (BM). J-alg was able to learn optimal strategies more quickly than M-Qubed and BM due to the Exploration stage (i.e., S'-alg). QL, QLOP, M-Qubed, and BM particularly gained low payoffs in the prisoner's dilemma (PD) game, because of the mutual defection (b, d) caused by many explorations to learn the cooperative joint action (a, c). QL and QLOP required many interactions, therefore they were not able to learn quickly in most of the games, but QLOP gained a higher average payoff than QL. QL and QLOP tended to obtain high payoffs

Table 3. Games used in the experiments

	c	*d*
a	***1.0,1.0***	0.0,0.0
b	0.0,0.0	0.5,0.5

(a) Common interest game (CIG)

	c	*d*
a	***1.0,0.5***	0.0,0.0
b	0.0,0.0	***0.5,1.0***

(b) Coordination game (CG)

	c	*d*
a	***1.0,1.0***	0.0,0.75
b	0.75,0.0	0.5,0.5

(c) Stag hunt (SH)

	c	*d*
a	0.0,1.0	***1.0,0.67***
b	0.33,0.0	0.67,0.33

(d) Tricky game (TG)

	c	*d*
a	***0.6,0.6***	0.0,1.0
b	1.0,0.0	0.2,0.2

(e) Prisoner's dilemma (PD)

	c	*d*
a	0.0,0.0	***0.67,1.0***
b	***1.0,0.67***	0.33,0.33

(f) Battle of the sexes (BS)

	c	*d*
a	***0.84,0.84***	0.33,1.0
b	1.0,0.33	0.0,0.0

(g) Chicken (Ch)

	c	*d*
a	0.84,0.33	0.84,0.0
b	0.0,1.0	***1.0,0.67***

(h) Security game (SG)

	c	*d*
a	0.0,0.0	***0.0,1.0***
b	***1.0,0.0***	0.0,0.0

(i) Offset game (OG)

	c	*d*
a	***1.0,0.0***	***0.0,1.0***
b	***0.0,1.0***	***1.0,0.0***

(j) Matching pennies (MP)

Table 4. Normalized average payoffs when agents played games in self-play. The bold typeface indicates the best results among the six given methods.

	QL	QLOP	M-Qubed	S-alg	BM	J-alg
CIG	0.76861	0.89883	0.99023	0.99846	0.99860	**0.99851**
CG	0.89430	0.88677	0.76217	0.99665	0.77492	**0.99679**
SH	0.71853	0.90423	0.99138	0.99853	0.99768	**0.99861**
TG	0.83351	0.84026	0.70782	**0.99719**	0.71618	0.99684
PD	0.53761	0.55262	0.68807	**0.99766**	0.69115	0.99760
BS	0.91072	0.91089	0.85186	0.99775	0.86266	**0.99776**
Ch	0.88359	0.93491	0.89158	**0.99826**	0.88656	0.99819
SG	0.69448	0.70582	0.83697	0.71804	0.84340	**0.99760**
OG	0.46086	0.43858	0.44924	**0.50012**	0.44855	0.45497
MP	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
Avg.	0.77022	0.80729	0.81693	0.92027	0.82197	**0.94369**

in the games that had two maximum joint actions. S-alg gained high payoffs in eight games, not including the security game (SG) and the offset game (OG). In these games, the aspiration level of S-alg decreased below the second-best payoff because of insufficient exploration. For all algorithms, the offset game (OG) proved difficult to learn the optimal strategy.

4.2 Experiment 2: Round-Robin

In our second experiment, we compared the six methods in a round-robin tournament in which all combinations of the methods, such as J-alg versus M-Qubed and S-alg versus BM, were examined. The agents played one of the 10 games for 50000 rounds, iterating 50 times for each game. In an asymmetric game, the agents played it twice with exchanging the roles (row and column) because the payoff changed by the role. The normalized average payoffs of each game are shown in Table 5. Values greater than one indicate that the agent exploited other agents and gained high payoffs as a result.

J-alg gained high average payoffs in many of the games. In the coordination game (CG), battle of the sexes (BS), the offset game (OG), and matching pennies (MP), S-alg was exploited when the associate took a greedy strategy, because its aspiration level decreased too much and it was satisfied with small payoffs as a result. QL and QLOP were not able to learn optimal strategies quickly and gain high average payoffs because they required a large number of interactions. QL was not so exploited by other players, compared to S-alg. In the prisoner's dilemma (PD), QL and QLOP gained much lower payoffs than the other four algorithms. In particular, QL gained the worst average payoffs because it learned slowly. Note that M-Qubed did indeed gain high average payoffs in the coordination game (CG), battle of the sexes (BS) and matching pennies (MP) by taking a greedy strategy.

Table 5. Normalized average payoffs when agents play games in a round-robin tournament. The bold typeface indicates the best results from among the six given methods.

	QL	QLOP	M-Qubed	S-alg	BM	J-alg
CIG	0.85129	0.90721	0.96661	0.95852	**0.96908**	0.96786
CG	0.89292	0.91256	**0.99598**	0.73174	0.96398	0.95050
SH	0.84249	0.91533	**0.96653**	0.95898	0.96137	0.96261
TG	0.85264	0.85388	0.87236	**0.91025**	0.86893	0.89840
PD	0.61984	0.62782	0.78704	0.76128	0.78030	**0.86004**
BS	0.90111	0.92201	**1.03072**	0.83428	0.99703	0.96876
Ch	0.88749	0.88233	0.93156	**0.94559**	0.93155	0.94303
SG	0.75910	0.76319	0.82802	0.84455	0.83136	**0.85919**
OG	0.43405	0.42567	0.59850	0.19922	0.66212	**0.66628**
MP	0.86232	0.89970	**1.16766**	0.75635	1.16150	1.15248
Avg.	0.79032	0.81097	0.91450	0.79008	0.91272	**0.92291**

The average payoffs shown in Tables 4 and 5 show that J-alg gained the highest average payoffs both in self-play and in the round-robin tournament. S-alg gained high payoffs in self-play, but it was exploited by greedy players and gained low payoffs in the round-robin tournament. M-Qubed and BM gained larger payoffs in the round-robin tournament than those in self-play, by exploiting S-alg, QL and QLOP. However, they required a large number of rounds to learn the maximum joint action and therefore gained lower average payoffs than that of J-alg. QL and QLOP required a large number of rounds to learn the maximum joint action both in self-play and in the round-robin tournament.

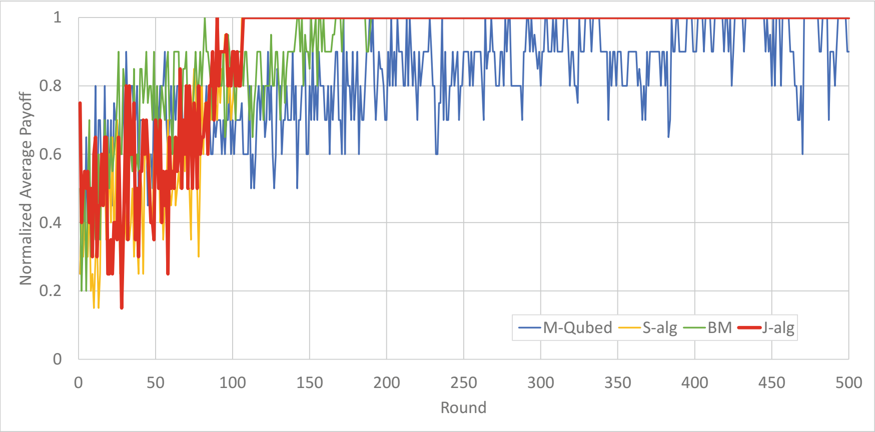
4.3 Learning Curves

Here, we investigated whether S'-alg actually contributed to the learning speed of J-alg. Figures 1, 2, and 3 show learning curves of the six methods in the common interest game (CIG), stag hunt (SH), and the security game (SG), respectively. Results of these three games are good examples that show how the algorithms comprised in J-alg complement one another.

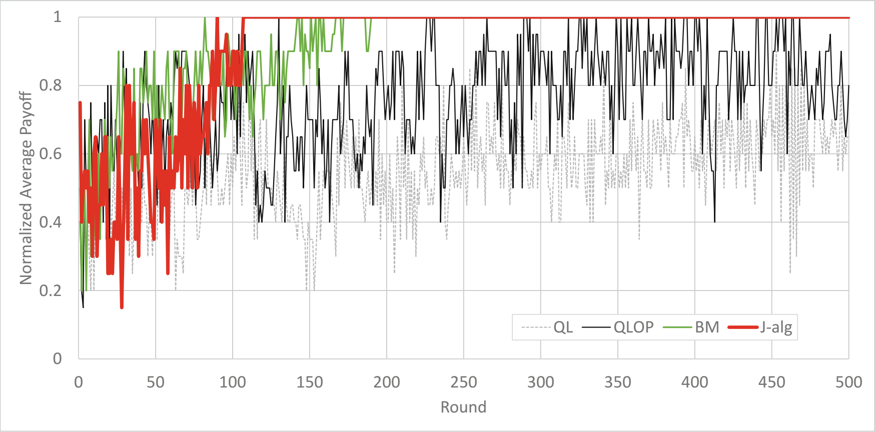
Figure 1, i.e., results of the common interest game (CIG), shows that J-alg and S-alg quickly learned the maximum joint action almost simultaneously. BM learned the maximum joint action by the 190th round, shortly after J-alg and S-alg. M-Qubed was still learning at the 500th round. Results here show that S'-alg contributed to the learning speed of J-alg. QL and QLOP required more interaction and were not able to finish learning by the 500th round.

Results of the stag hunt (SH) game are shown in Fig. 2, which shows that J-alg and S-alg finished learning the maximum joint action the quickest. BM finished learning with the normalized average payoff converging to one by the 410th round. M-Qubed was still learning at the 1500th round. Results here show the same as Fig. 1, i.e., J-alg learned the maximum joint action fastest, whereas BM and M-Qubed learned more slowly. QL and QLOP required more interaction and were not able to finish learning by the 1500th round.

Turning to Fig. 3, we observe that in the security game (SG), S-alg finished learning first, but did not obtain the optimal strategy because S-alg was satisfied with the second-best payoff. As shown in Table 4, S-alg gained smaller payoffs as compared with M-Qubed, BM, and J-alg in this game. BM and M-Qubed learned the maximum joint action, but they required a large number of interactions. QL and QLOP were worse than S-alg. Despite the failure of S-alg, J-alg successfully learned the maximum joint action the quickest in this game.

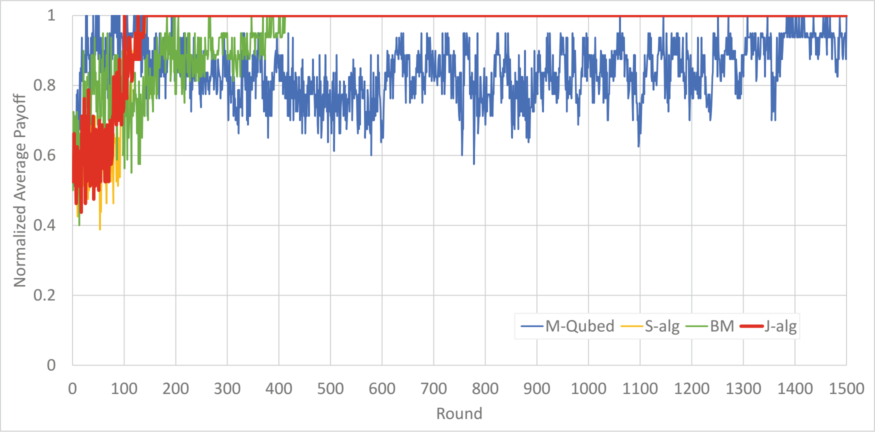


(a) Comparing J-alg with M-Qubed, S-alg, and BM

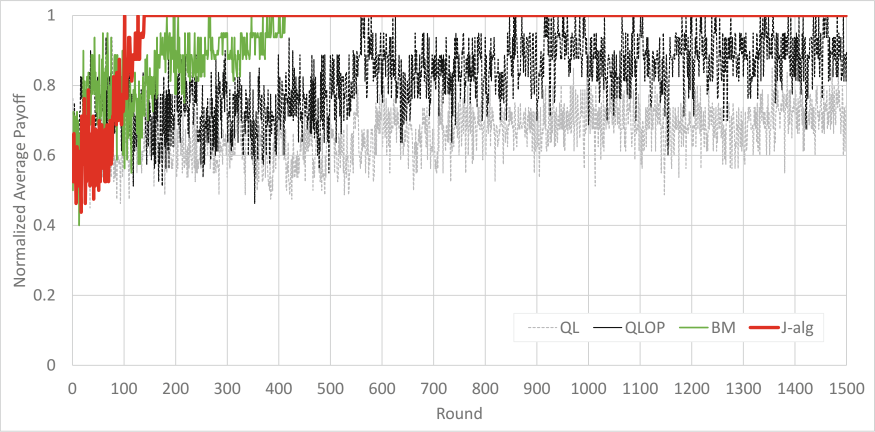


(b) Comparing J-alg with QL, QLOP, and BM

Fig. 1. Learning curves of the six algorithms in self-play in Common Interest Game (CIG). Red: the curve of J-alg. Blue: that of M-Qubed. Yellow: that of S-alg. Green: that of BM. Gray: that of QL. Black: that of QLOP. The x-axes show the rounds the agents played, while the y-axes show the normalized average payoffs. (Color figure online)

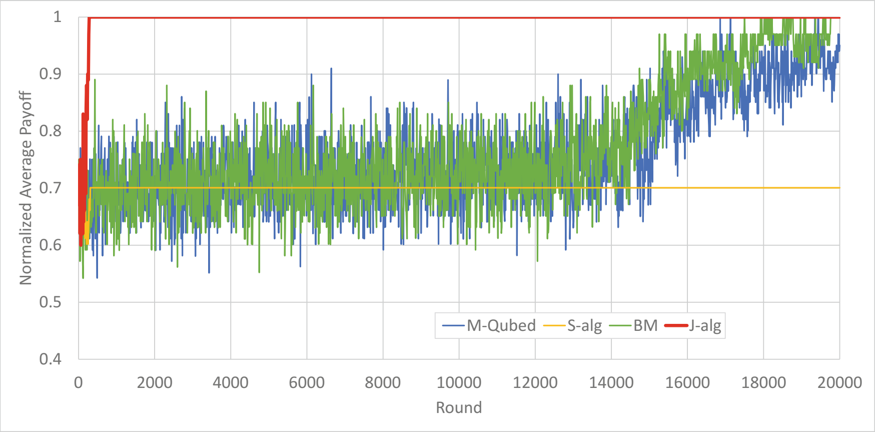


(a) Comparing J-alg with M-Qubed, S-alg, and BM

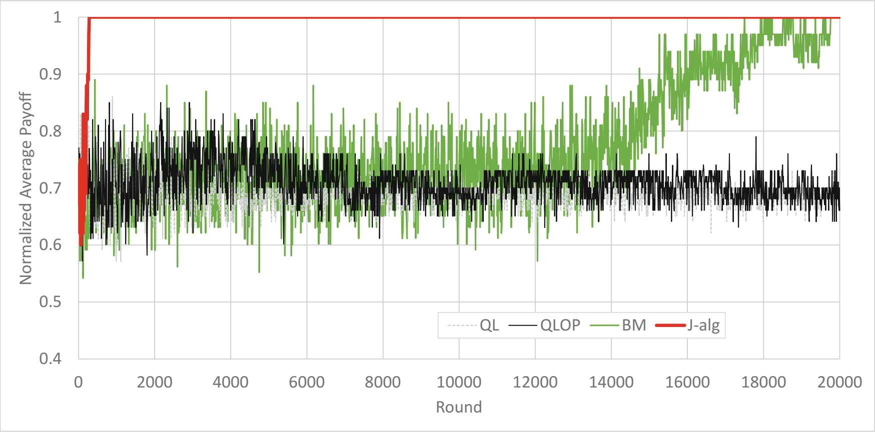


(b) Comparing J-alg with QL, QLOP, and BM

Fig. 2. Learning curves of the six algorithms in self-play in Stag Hunt (SH). Red: the curve of J-alg. Blue: that of M-Qubed. Yellow: that of S-alg. Green: that of BM. Gray: that of QL. Black: that of QLOP. The x-axes show the rounds the agents played, while the y-axes show normalized average payoffs. (Color figure online)



(a) Comparing J-alg with M-Qubed, S-alg, and BM



(b) Comparing J-alg with QL, QLOP, and BM

Fig. 3. Learning curves of the six algorithms in self-play in Security Game (SG). Red: the curve of J-alg. Blue: that of M-Qubed. Yellow: that of S-alg. Green: that of BM. Gray: that of QL. Black: that of QLOP. The x-axes show the rounds the agents played, while the y-axes show normalized average payoffs. (Color figure online)

5 Discussion

The most remarkable feature of J-alg is that it has two learning stages. Schembri et al. [9] proposed a learning robot that has two stages named “childhood” and “adulthood”. During childhood, the robot learns basic movements, and it learns how to combine the basic movements in order to perform a task during adulthood. To success of the learning, it is important how to learn in the early stage of learning.

J-alg uses S’-alg, a slightly modified version of S-alg, in the Exploration stage that allows the agent to quickly learn the cooperative action against the cooperative associate. On the other hand, failure of learning the maximum joint action in S-alg leads the fact that the associate is not always cooperative but may be a greedy player or simply a slow learner. To such associates, BM carefully learns its strategy in the Static stage.

Success of J-alg in the self-play tournament is derived from the fact that J-alg agents take a cooperative strategy from the beginning of games. M-Qubed, BM, and QLOP are different from J-alg in the initial phase of learning. Since their Q-values are set optimistically, they play randomly for a long time until convergence. The contribution of S’-alg in the early learning phase allowed J-alg to cooperate with each other and gain high payoffs from the beginning.

Success of J-alg in the round-robin tournament is derived from the fact that J-alg agents are resistant to exploitation by greedy associates but willing to cooperate against cooperative associates. To greedy associates, J-alg agents take similar behavior of BM in their long life, but to cooperative associates, different from BM, they are “innocent” and able to cooperate with them from the beginning.

In the environment where other players also learn, it may be a good idea that an agent first takes a cooperative, innocent strategy but a cautious, suspicious strategy later, like children grow to adults. This study suggests that this idea is plausible by showing that such approach can acquire high payoffs in various situations.

6 Related Works

Reinforcement learning is widely used as a learning method for agents. What defines reinforcement learning as being different from other learning methods is the combination of trial-and-error searches and delayed rewards. Temporal difference (TD) learning is arguably the most famous reinforcement learning method. TD learning compares the value of the current state with the reward obtained by the current action, and updates the value to decrease deviation. Q-learning [12] and Sarsa [8] are typical examples of TD learning.

While TD learning was originally proposed as a learning method for a single agent, Claus and Boutilier [2] discussed how to incorporate other agents in a multi-agent environment in TD learning. They showed that in payoff matrix games, a “joint action learner” that updates Q-values based on actions of the agent and others is better than an “independent learner” that recognizes others as elements of the environment.

When designing a reinforcement learning agent in a multi-agent environment, we often adopt the knowledge of game theory. Hu and Wellman [5] proposed Nash Q-learning. A Nash Q-learning agent explores the game structure from the perspective of the actions and rewards of the agent and others, updating its Q-values under the assumption that every player selects a strategy that leads to a Nash equilibrium.

There are also methods that do not update Q-values from immediate joint actions, but instead assign Q-values to a long (fixed-length) interaction history. Burkov and Chaib-draa [1] proposed a learning approach for an agent when its associates learn its strategies and adapt to its actions. Their agent, called an Adaptive Dynamics Learner, can obtain higher utility than that of an agent at an equilibrium by considering the associates' strategies and assigning Q-values to a long interaction history.

Further, there are reinforcement learning methods that do not use Q-values, instead they use a dynamic threshold called an aspiration level updated by rewards. Masuda and Nakamura [6] considered a model in which a reinforcement learning agent learns from reinforcement signals calculated from the reward and the aspiration level; they investigated agent behavior when it played the iterated prisoner's dilemma games with other learning agents.

In summary, many researchers have proposed reinforcement learning algorithms for multi-agent environments; however, much of this work has focused only on the convergence of player strategies and thus requires an enormous number of interactions. In this paper, we proposed an algorithm that learns appropriate strategies against the associate's strategy faster than these existing algorithms. It is necessary for agents in the real world because they have to decide many things very quickly there.

7 Conclusions

Many researchers have studied and are studying reinforcement learning algorithms to acquire strategies that maximize payoffs for agents that learn in games. Existing reinforcement learning algorithms can gain high payoffs in various games, but typically require a large number of interactions to learn an optimal strategy.

To overcome these limitations, we proposed an algorithm called J-alg that learns quickly and gains large payoffs in many games by combining two existing algorithms, namely, BM and S'-alg. S'-alg contributes to learning speed, whereas BM contributes to the prevention of exploitation by greedy opponents. At the beginning, J-alg starts to learn the maximum joint action with the associate with S'-alg that prefers a cooperative strategy. After that, in the latter stage where BM takes control, if the maximum joint action is realized, BM continues to take the action, but if not, BM carefully learns against (greedy) opponents.

To evaluate our algorithm, we conducted two experiments, i.e., self-play and round-robin, using 10 games. In both experiments, J-alg gained sufficiently large payoffs in nine games and the highest average payoffs among six types of agents. Dividing learning into two stages has led to the success of J-alg.

As future work, we plan to construct an algorithm that can learn the maximum joint action in the offset game (OG). In addition, we plan to extend our work in two-person two-action games and construct an algorithm that can quickly learn optimal strategies in n-person n-action games.

References

1. Burkov, A., Chaib-draa, B.: Effective learning in the presence of adaptive counterparts. J. Algorithm. **64**, 127–138 (2009)
2. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the 15th National Conference on Artificial Intelligence (AAAI), pp. 746–752 (1998)
3. Crandall, J.W., Goodrich, M.A.: Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning. Mach. Learn. **82**, 281–314 (2011)
4. Fujita, W., Moriyama, K., Fukui, K., Numao, M.: Learning better strategies with a combination of complementary reinforcement learning algorithms. In: Nishizaki, S., Numao, M., Caro, J.D.L., Suarez, M.T.C. (eds.) Theory and Practice of Computation, pp. 43–54. World Scientific, Singapore (2016)
5. Hu, J., Wellman, M.P.: Nash Q-learning for general-sum stochastic games. J. Mach. Learn. Res. **4**, 1039–1069 (2003)
6. Masuda, N., Nakamura, M.: Numerical analysis of a reinforcement learning model with the dynamic aspiration level in the iterated Prisoner's dilemma. J. Theor. Biol. **278**, 55–62 (2011)
7. Okada, A.: Game Theory, New edn. Yuhikaku, Tokyo (2011). (in Japanese)
8. Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems. Technical report TR166, Cambridge University Engineering Department (1994)
9. Schembri, M., Mirolli, M., Baldassarre, G.: Evolving internal reinforcers for an intrinsically motivated reinforcement-learning robot. In: Proceedings of the 6th IEEE International Conference on Development and Learning (ICDL), pp. 282–287 (2007)
10. Stimpson, J.L., Goodrich, M.A.: Learning to cooperate in a social dilemma: a satisficing approach to bargaining. In: Proceedings of the 20th International Conference on Machine Learning (ICML), pp. 728–735 (2003)
11. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
12. Watkins, C.J.C.H., Dayan, P.: Technical note: Q-learning. Mach. Learn. **8**, 279–292 (1992)
13. Wiering, M.A., van Hasselt, H.: Ensemble algorithms in reinforcement learning. IEEE Trans. Syst. Man Cybern. B **38**, 930–936 (2008)

Recursive Reductions of Action Dependencies for Coordination-Based Multiagent Planning

Jan Tožička[1], Jan Jakubův[2], and Antonín Komenda[1](✉)

[1] Department of Computer Science, Artificial Intelligence Center,
Czech Technical University in Prague,
Karlovo namesti 13, 121 35 Prague, Czech Republic
{jan.tozicka,antonin.komenda}@agents.fel.cvut.cz
[2] Intelligent Systems, Czech Institute of Informatics,
Robotics, and Cybernetics, Czech Technical University in Prague,
Zikova Street 1903/4, 166 36 Prague 6, Czech Republic
jakubja5@ciirc.cvut.cz

Abstract. Currently the most efficient distributed multiagent planning scheme for deterministic models is based on coordination of local agents' plans. In such a scheme, behavior of other agents is modeled using projections of their actions stripped of all private information. The planning scheme does not require any additional information, however using such can be beneficial for planning efficiency. Dependencies among the projected public actions caused by sequences of local private actions represent one particular type of such information.

In this work, we formally define several types of internal dependencies of multiagent planning problems and provide an algorithmic approach how to extract the internally dependent actions during multiagent planning. We show how to take an advantage of the computed dependencies by means of reducing the multiagent planning problems and analyze worst-case privacy leakage caused by the used dependencies. We integrate the reduction method into a distributed multiagent planner and summarize other efficiency improving techniques used in the planner. We experimentally show strong reduction of majority of standard multiagent benchmarks and nearly doubling of solved problems in comparison to a variant of a planner without the reductions. The efficiency of the method is demonstrated by winning in a recent competition of distributed multiagent planners.

1 Introduction

Cooperative intelligent agents acting in a shared environment have to coordinate their steps in order to achieve their goals. A well-established model for multiagent planning in deterministic environments was described by [4] as MA-Strips, which is a minimal extension of classical planning model Strips [8]. MA-Strips provides problem partitioning in form of separated sets of actions of particular

© Springer International Publishing AG, part of Springer Nature 2018
N. T. Nguyen et al. (Eds.): TCCI XXVIII, LNCS 10780, pp. 66–92, 2018.
https://doi.org/10.1007/978-3-319-78301-7_4

agents, and notion of local private information the agents are not willing to share. By definition, private actions and facts about the environment do not affect other agents and cannot be affected by other agents. Shared facts and actions which can influence more than one agent are denoted as public.

In multiagent planning modeled as MA-Strips, agents can either plan only with their own actions and facts and inform the other agents about public achieved facts, as for instance in the MAD-A* planner [16]. Or, agents can also use other agents' public actions provided that the actions are stripped of the private facts in preconditions and effects. Thus agents plan actions, in a sense, for other agents and then coordinate the plans [19]. This principle was successfully used in a recent planner based on compact representation of local agents' plans in form of Finite Automata, denoted as Planning State Machines (Psm) planner [21].

Only a complete stripping of all private information from public actions was used in literature so far. Such approach can, however, lead to tangible loss of information on causal dependencies of the actions described by the private actions. A seeming remedy is to borrow techniques from classical planning on problem reduction (e.g., in [5,7,9]). As our motivation is to "pack" sequences of public and private actions, the most suitable are recursive macro actions as proposed by [1,13]. A macro action can represent a sound sequence of actions and, provided that it allows for recursive reductions, it can be used repeatedly with possibly radical downsizing of the reduced planning problem.

In a motivation logistic problem, when an agent transports a package from one city to another and wants to keep its current load internal, it is not practical to publish two actions: *load(package, fromCity)* and *unload(package, toCity)*. Instead it should publish action *transport(package, fromCity, toCity)*, which is capturing the hidden (private) relation between this pair of actions while it is not disclosing it in an explicit way.

We propose to keep the pair of actions and to add new public predicate that says that some action requires another action to precede it (because it better fits the proposed coordination algorithm). In the simplest case, the new public fact would directly correspond to the internal fact *isLoaded(package)*, but in realistic cases, it could also capture more complex dependencies, for example, the transshipment between different vehicles belonging to the transport agent.

In this paper, we build on our previous work [17] where internal dependencies of public actions where studied with a restriction that every action consumes all of its preconditions. This restriction no longer applies here because it can be very limiting in practice. Furthermore, we demonstrate the effect of our approach on a benchmark set from the CoDMAP competition held on International Conference on Automated Planning and Scheduling (ICAPS'15). Our planner, employing the theory from this paper, won the distributed track of CoDMAP 2015.

2 Multiagent Planning

This section provides condensed formal prerequisites of multiagent planning based on the MA-Strips formalism [4]. Refer to [18] for more details.

An MA-Strips *planning problem* $\mathit{\Pi}$ is a quadruple $\mathit{\Pi} = \langle P, \{\alpha_i\}_{i=1}^{n}, I, G\rangle$, where P is a set of facts, α_i is the set of actions of i-th agent[1] $I \subseteq P$ is an initial state, and $G \subseteq P$ is a set of goal facts. We define selector functions $\mathsf{facts}(\mathit{\Pi})$, $\mathsf{agents}(\mathit{\Pi})$, $\mathsf{init}(\mathit{\Pi})$, and $\mathsf{goal}(\mathit{\Pi})$ such that $\mathit{\Pi} = \langle \mathsf{facts}(\mathit{\Pi}), \mathsf{agents}(\mathit{\Pi}), \mathsf{init}(\mathit{\Pi}), \mathsf{goal}(\mathit{\Pi})\rangle$. An *action* $a \in \alpha$, the agent α can perform, is a triple of subsets of P called *preconditions*, *add effects*, *delete effects*. Selector functions $\mathsf{pre}(a)$, $\mathsf{add}(a)$, and $\mathsf{del}(a)$ are defined so that $a = \langle \mathsf{pre}(a), \mathsf{add}(a), \mathsf{del}(a)\rangle$. Note that an agent is identified with the actions the agent can perform in an environment.

In MA-Strips, out of computational or privacy concerns, each fact is classified either as *public* or as *internal*. A fact is *public* when it is mentioned by actions of at least two different agents. A fact is *internal for agent* α when it is not public but mentioned by some action of α. A fact is *relevant for* α when it is either public or internal for α. MA-Strips further extends this classification of facts to actions as follows. An action is *public* when it has a public (add- or delete-) effect, otherwise it is *internal*. An action from $\mathit{\Pi}$ is *relevant for* α when it is either public or owned by (contained in) α.

We use $\mathsf{int\text{-}facts}(\alpha)$ and $\mathsf{pub\text{-}facts}(\alpha)$ to denote in turn the sets internal facts and the set of public facts of agent α. Moreover, we write $\mathsf{pub\text{-}facts}(\mathit{\Pi})$ to denote all the public facts of problem $\mathit{\Pi}$. We write $\mathsf{pub\text{-}actions}(\alpha)$ to denote the set of public actions of agent α. Finally, we use $\mathsf{pub\text{-}actions}(\mathit{\Pi})$ to denote all the public actions of all the agents in problem $\mathit{\Pi}$.

In multiagent planning with external actions, a *local planning problem* is constructed for every agent α. Each local planning problem for α is a classical Strips problem where α has its own internal copy of the global state and where each agent is equipped with information about public actions of other agents called *external actions*. These local planning problems allow us to divide an MA-Strips problem into several Strips problems which can be solved separately by a classical planner.

The *projection* $F \triangleright \alpha$ of a set of facts F to agent α is the restriction of F to the facts relevant for α, representing F as seen by α. The *public projection* $a \triangleright \star$ of action a is obtained by restricting the facts in a to public facts. Public projection is extended to sets of actions element-wise.

A *local planning problem* $\mathit{\Pi} \triangleright \alpha$ of agent α, also called *projection of* $\mathit{\Pi}$ to α, is a classical Strips problem containing all the actions of agent α together with external actions, that is, public projections of other agents' public actions. The local problem of α is defined only using the facts relevant for α. Formally,

$$\mathit{\Pi} \triangleright \alpha = \langle P \triangleright \alpha, \alpha \cup \mathsf{exts}(\alpha), I \triangleright \alpha, G\rangle$$

where the set of external actions $\mathsf{exts}(\alpha)$ is defined as follows.

$$\mathsf{exts}(\alpha) = \bigcup_{\beta \neq \alpha} (\mathsf{pub\text{-}actions}(\beta) \triangleright \star)$$

---

[1] Whereas, in Strips, the second parameter is a set of actions, in MA-Strips, the second parameter is actually a set of sets of actions.

In the above, β ranges over all the agents of Π. The set $\mathsf{exts}(\alpha)$ can be equivalently described as $\mathsf{exts}(\alpha) = (\mathsf{pub\text{-}actions}(\Pi) \setminus \alpha) \triangleright \star$. To simplify the presentation, we consider only problems with public goals and hence there is no need to restrict goal G.

3 Planning with External Actions

The previous section allows us to divide an MA-Strips problem into several classical Strips local planning which can be solved separately by a classical planner. Recall that the local planning problem of agent α contains all the actions of α together with α's external actions, that is, with projections of public actions of other agents. This section describes conditions which allow us to compute a solution of the original MA-Strips problem from solutions of local problems.

A *plan* π is a sequence of actions. A *solution* of Π is a plan π whose execution transforms the initial state to a subset of the goals. A *local solution* of agent α is a solution of $\Pi \triangleright \alpha$. Let $\mathsf{sols}(\Pi)$ denote the set of all the solutions of MA-Strips or Strips problem Π. A *public plan* σ is a sequence of public actions. The *public projection* $\pi \triangleright \star$ *of plan* π is the restriction of π to public actions.

A public plan σ is *extensible* when there is $\pi \in \mathsf{sols}(\Pi)$ such that $\pi \triangleright \star = \sigma$. Similarly, σ is α*-extensible* when there is $\pi \in \mathsf{sols}(\Pi \triangleright \alpha)$ such that $\pi \triangleright \star = \sigma$. Extensible public plans give us an order of public actions which is acceptable for all the agents. Thus extensible public plans are very close to solutions of Π and it is relatively easy to construct a solution of Π once we have an extensible public plan. Hence our algorithms will aim at finding extensible public plans.

The following theorem [18] establishes the relationship between extensible and α-extensible plans. Its direct consequence is that to find a solution of Π it is enough to find a local solution $\pi_\alpha \in \mathsf{sols}(\Pi \triangleright \alpha)$ which is β-extensible for every agent β.

Theorem 1. *Public plan* σ *of* Π *is extensible if and only if* σ *is* α*-extensible for every agent* α*.*

The theorem above suggests the distributed multiagent planning algorithm described in Algorithm 1. Every agent executes the loop from Algorithm 1, possibly on different machine. Every agent keeps generating new solutions of its local problem and stores solution projections in set Φ_α. These sets are exchanged among all the agents so that every agent can compute their intersection Φ. Once the intersection Φ is non-empty, the algorithm terminates yielding Φ as the result. Theorem 1 ensures that every public plan in the resulting Φ is extensible. See Sect. 8 for more details on the algorithm implementation.

4 Internal Dependencies of Actions

One of the benefits of planning with external actions is that every agent can plan separately its local problem which involves planning of actions for other agents

Algorithm 1. Distributed MA planning algorithm.

```
Function MaPlanDistributed($\Pi \triangleright \alpha$) is
    $\Phi_\alpha \leftarrow \emptyset$;
    loop
        generate new $\pi_\alpha \in \mathsf{sols}(\Pi \triangleright \alpha)$;
        $\Phi_\alpha \leftarrow \Phi_\alpha \cup \{\pi_\alpha \triangleright \star\}$;
        exchange public plans $\Phi_\alpha$ with other agents;
        $\Phi \leftarrow \bigcap_{\beta \in \mathsf{agents}(\Pi)} \Phi_\beta$;
        if $\Phi \neq \emptyset$ then
            return $\Phi$;
        end
    end
end
```

(external actions). Other agents can then only verify whether a plan generated by another agent is α-extensible for them. A con of this approach is that agents have only a limited knowledge about external actions because internal facts are removed by projection. Thus it can happen that an agent plans external actions inappropriately in a way that the resulting public plan is not α-extensible for some agent α.

In the rest of this paper we try to overcome the limitation of partial information about external actions. The idea is to equip agents with additional information about external actions without revealing internal facts. The rest of this section describes *dependency graphs* which are used in the following sections as a formal ground for our analysis of public and external actions.

4.1 Dependency Graphs

Local planning problem $\Pi \triangleright \alpha$ of agent α contains information about external actions provided by the set $\mathsf{exts}(\alpha)$. The idea is to equip agent α with more information described by a suitable structure. A dependency graphs is a structure we use to encapsulate information about public actions which an agent shares with other agents.

Dependency graphs are known from literature [6,14]. In our context, a *dependency graph* Δ is a bipartite directed graph defined as follows.

Definition 1. *A* dependency graph Δ *is a bipartite directed graph whose nodes are actions and facts. We write* $\mathsf{actions}(\Delta)$ *and* $\mathsf{facts}(\Delta)$ *to denote action and fact nodes respectively. Given the nodes, graph* Δ *contains the following three kinds of edges.*

$$\begin{array}{lll} (a \rightarrow f) \in \Delta & \textit{iff } f \in \mathsf{add}(a) & \textit{(a produces f)} \\ (f \rightarrow a) \in \Delta & \textit{iff } f \in \mathsf{pre}(a) \setminus \mathsf{del}(a) & \textit{(a requires f)} \\ (f \dashrightarrow a) \in \Delta & \textit{iff } f \in \mathsf{pre}(a) \cap \mathsf{del}(a) & \textit{(a consumes f)} \end{array}$$

Additionally, a fact can be marked as initial in Δ. The set of states marked as initial is denoted $\mathsf{init}(\Delta)$.

Hence edges of a dependency graph Δ are uniquely determined by the set of nodes. Note that action nodes are themselves actions, that is, triples of fact sets. These action nodes can contain additional facts other than fact nodes $\mathsf{facts}(\Delta)$. We use dependency graphs to represent internal dependencies of public actions. Dependencies determined by public facts are known to other agents and thus we do not need them in the graph as fact nodes. From now on we suppose that $\mathsf{facts}(\Delta)$ contains no public facts as fact nodes. Action nodes, however, can contain public facts in their public actions.

Definition 2. *Let an* MA-STRIPS *problem Π be given. The* minimal dependency graph $\mathsf{MD}(\alpha)$ *of agent $\alpha \in$* $\mathsf{agents}(\Pi)$ *is the dependency graph uniquely determined by the following set of nodes.*

$$\begin{aligned}\mathsf{actions}(\mathsf{MD}(\alpha)) &= \mathsf{pub\text{-}actions}(\alpha)\\ \mathsf{facts}(\mathsf{MD}(\alpha)) &= \emptyset\\ \mathsf{init}(\mathsf{MD}(\alpha)) &= \emptyset\end{aligned}$$

Hence $\mathsf{MD}(\alpha)$ has no edges as there are no fact nodes. Thus the graph contains only separated public action nodes. Furthermore, the set $\mathsf{exts}(\alpha)$ of external actions of agent α can be trivially expressed as follows.

$$\mathsf{exts}(\alpha) = \bigcup_{\beta \neq \alpha} (\mathsf{actions}(\mathsf{MD}(\beta)) \rhd \star)$$

Thus we see that dependency graphs can carry the same information as provided by $\mathsf{exts}(\alpha)$.

Definition 3. *The* full dependency graph $\mathsf{FD}(\alpha)$ *of agent α contains all the actions of α and all the internal facts of α.*

$$\begin{aligned}\mathsf{actions}(\mathsf{FD}(\alpha)) &= \alpha\\ \mathsf{facts}(\mathsf{FD}(\alpha)) &= \mathsf{int\text{-}facts}(\alpha)\\ \mathsf{init}(\mathsf{FD}(\alpha)) &= \mathsf{init}(\Pi) \cap \mathsf{int\text{-}facts}(\alpha)\end{aligned}$$

Hence $\mathsf{FD}(\alpha)$ contains all the information known by α. By publishing $\mathsf{FD}(\alpha)$, an agent reveals all his internal dependencies which might be a potential privacy risk. On the other hand, other agents are by $\mathsf{FD}(\alpha)$ provided the most precise information about dependencies of public actions of α. Every plan of another agent, computed with $\mathsf{FD}(\alpha)$ in mind, is automatically α-extensible. Thus we see that dependency graphs can carry dependencies information with a varied precision.

4.2 Dependency Graph Collections

A dependency graph represents information about public actions of one agent. Every agent needs to know information from all the other agents. We use *dependency graph collections* to represent all the required information. A *dependency* *graph collection* $\mathcal{D}$ of an MA-Strips problem Π is a set of dependency graphs which contains exactly one dependency graph for every agent of Π. We write $\mathcal{D}(\alpha)$ to denote the graph of α. We write $\mathsf{actions}(\mathcal{D})$, $\mathsf{facts}(\mathcal{D})$, and $\mathsf{init}(\mathcal{D})$ to denote in turn all the action, fact, and initial fact nodes from all the graphs in $\mathcal{D}$.

Definition 4. *Given problem* Π*, we can define the* minimal collection $\mathsf{MD}(\Pi)$ *and the* full collection $\mathsf{FD}(\Pi)$ *as follows.*

$$\begin{aligned}\mathsf{MD}(\Pi) &= \{\mathsf{MD}(\alpha) : \alpha \in \mathsf{agents}(\Pi)\}\\ \mathsf{FD}(\Pi) &= \{\mathsf{FD}(\alpha) : \alpha \in \mathsf{agents}(\Pi)\}\end{aligned}$$

Later we shall show some interesting properties of the minimal and full collections.

4.3 Local Problems and Dependency Collections

In order to define local problems informed by $\mathcal{D}$, we need to define facts and action projections which preserve information from $\mathcal{D}$. We use symbol $\triangleright_{\mathcal{D}}$ to denote *projections accordingly to* $\mathcal{D}$. Recall that the public projection $a \triangleright \star$ of action a is the restriction of the facts of a to $\mathsf{pub\text{-}facts}(\Pi)$. The *public projection* $a \triangleright_{\mathcal{D}} \star$ *of action* a *accordingly to* $\mathcal{D}$ is the restriction of the facts of a to $\mathsf{pub\text{-}facts}(\Pi) \cup \mathsf{facts}(\mathcal{D})$. Public projection is extended to sets of actions element-wise. Furthermore, *external actions of* α *according to* $\mathcal{D}$, denoted $\overline{\mathsf{exts}}_{\mathcal{D}}(\alpha)$, contain public projections (according to $\mathcal{D}$) of actions of other agents. In other words, $\overline{\mathsf{exts}}_{\mathcal{D}}(\alpha)$ carries all the information published by other agents for agent α. It is computed as follows.

$$\overline{\mathsf{exts}}_{\mathcal{D}}(\alpha) = \bigcup_{\beta \neq \alpha} (\mathsf{actions}(\mathcal{D}(\beta)) \triangleright_{\mathcal{D}} \star)$$

This equation captures distributed computation of $\overline{\mathsf{exts}}_{\mathcal{D}}(\alpha)$ where every agent β separately computes published actions, applies public projection, and sends the result to α.

In order to define a local planning problem of agent α which would take information from $\mathcal{D}$ into consideration, we need to extract from $\mathcal{D}$ facts and initial facts of other agents. Below we define sets $\overline{\mathsf{facts}}_{\mathcal{D}}(\alpha)$ and $\overline{\mathsf{init}}_{\mathcal{D}}(\alpha)$ which contain those facts and initial facts published by other agents, that is, all the facts from $\mathcal{D}$ except of the facts of α.

$$\begin{aligned}\overline{\mathsf{facts}}_{\mathcal{D}}(\alpha) &= \mathsf{facts}(\mathcal{D}) \setminus \mathsf{facts}(\mathcal{D}(\alpha))\\ \overline{\mathsf{init}}_{\mathcal{D}}(\alpha) &= \mathsf{init}(\mathcal{D}) \setminus \mathsf{init}(\mathcal{D}(\alpha))\end{aligned}$$

Now we are ready to define *local planning problems according to* $\mathcal{D}$ which extends local planning problems by the information contained in $\mathcal{D}$.

Definition 5. *Let* Π *be* MA-Strips *problem. The* local problem $\Pi \triangleright_{\mathcal{D}} \alpha$ of agent $\alpha \in \mathsf{agents}(\Pi)$ accordingly to $\mathcal{D}$ *is the classical* Strips *problem* $\Pi \triangleright_{\mathcal{D}} \alpha = \langle P_0, A_0, I_0, G_0 \rangle$ *where*

(1) $P_0 = \mathsf{facts}(\Pi \triangleright \alpha) \cup \overline{\mathsf{facts}}_{\mathcal{D}}(\alpha)$,
(2) $A_0 = \alpha \cup \overline{\mathsf{exts}}_{\mathcal{D}}(\alpha)$,
(3) $I_0 = \mathsf{init}(\Pi \triangleright \alpha) \cup \overline{\mathsf{init}}_{\mathcal{D}}(\alpha)$, and
(4) $G_0 = \mathsf{goal}(\Pi)$.

We can see that a local problem $\Pi \triangleright_{\mathcal{D}} \alpha$ according to $\mathcal{D}$ extends the local problem $\Pi \triangleright \alpha$ by the facts and actions published by $\mathcal{D}$.

Let us consider two boundary cases of dependency collections $\mathsf{MD}(\Pi)$ and $\mathsf{FD}(\Pi)$. Given an MA-Strips problem Π, we can construct local problems using the minimal dependency collection $\mathsf{MD}(\Pi)$. It is easy to see that $\Pi \triangleright_{\mathsf{MD}(\Pi)} \alpha = \Pi \triangleright \alpha$ for every agent α. With the full dependency collection $\mathsf{FD}(\Pi)$ we obtain equal projections, that is, $\Pi \triangleright_{\mathsf{FD}(\Pi)} \alpha = \Pi \triangleright_{\mathsf{FD}(\Pi)} \beta$ for all agents α and β. Moreover, local solutions equal MA-Strips solutions, that is, $\mathsf{sols}(\Pi \triangleright_{\mathsf{FD}(\Pi)} \alpha) = \mathsf{sols}(\Pi)$ for every α.

4.4 Publicly Equivalent Problems

We have seen that dependency collections can provide information about internal dependencies with a varied precision. Given two different collections, two different local problems can be constructed for every agent. However, when the two local problems of the same agent equal on public solutions, we can say that they are equivalent because their public solutions are equally extensible.

In order to define equivalent collections, we first define public equivalence on problems. Two planning problems Π_0 and Π_1 are *publicly equivalent*, denoted $\Pi_0 \simeq \Pi_1$, when they have equal public solutions. Formally as follows.

$$\Pi_0 \simeq \Pi_1 \quad \Leftrightarrow \quad \mathsf{sols}(\Pi_0) \triangleright \star = \mathsf{sols}(\Pi_1) \triangleright \star$$

Public equivalence can be extended to dependency graph collections as follows. Two collections $\mathcal{D}_0$ and $\mathcal{D}_1$ of the same MA-Strips problem Π are *equivalent*, written $\mathcal{D}_0 \simeq \mathcal{D}_1$, when for any agent α, it holds that the local problems $\Pi \triangleright_{\mathcal{D}_0} \alpha$ and $\Pi \triangleright_{\mathcal{D}_1} \alpha$ are publicly equivalent. Formally as follows.

$$\mathcal{D}_0 \simeq \mathcal{D}_1 \quad \Leftrightarrow \quad (\Pi \triangleright_{\mathcal{D}_0} \alpha) \simeq (\Pi \triangleright_{\mathcal{D}_1} \alpha) \ \ \text{(for all } \alpha\text{)}$$

Example 1. Given an MA-Strips problem Π, with the full dependency collection $\mathsf{FD}(\Pi)$ we can see that $\Pi \simeq \Pi \triangleright_{\mathsf{FD}(\Pi)} \alpha$ holds for any agent. Hence to find a public solution of Π it is enough to solve the local problem (accordingly to $\mathsf{FD}(\Pi)$) of an arbitrary agent. The same holds for any dependency collection $\mathcal{D}$ such that $\mathcal{D} \simeq \mathsf{FD}(\Pi)$. Note that $\mathcal{D}$ can be much smaller and provide less private information than the full dependency collection.

The above definitions allow us to recognize problems without any internal dependencies which we can define as follow.

Definition 6. *An* MA-Strips *problem Π is* internally independent *when*

$$\mathsf{MD}(\Pi) \simeq \mathsf{FD}(\Pi).$$

In order to solve an internally independent problem, it is enough to solve the local problem $\Pi \triangleright \alpha$ of an arbitrary agent. Any local public solution is extensible which makes internally independent problems easier to solve because there is no need for interaction and negotiation among the agents. Later we shall show how to algorithmically recognize internally independent problems. The following formally captures the above properties.

Lemma 1. *Let Π be an internally independent* MA-Strips *problem. Then* $(\Pi \triangleright \alpha) \simeq \Pi$.

Proof. $(\Pi \triangleright \alpha) \simeq (\Pi \triangleright_{\mathsf{MD}(\Pi)} \alpha) \simeq (\Pi \triangleright_{\mathsf{FD}(\Pi)} \alpha) \simeq \Pi$.

5 Simple Action Dependencies

Let us consider dependency collections without internal actions, that is, collections $\mathcal{D}$ where $\mathsf{actions}(\mathcal{D})$ contains no internal actions. When $\mathcal{D}$ is published, then no agent publishes actions additional to $\mathsf{exts}(\alpha)$ which is desirable out of privacy concerns. Furthermore, the plan search space of $\Pi \triangleright_{\mathcal{D}} \alpha$ is not increased when compared to $\Pi \triangleright \alpha$. Even more, every additionally published fact in $\mathcal{D}$ providing a valid dependency prunes the search space. Action dependencies captured by collections without internal actions can be expressed by requirements on the order of actions in a plan. This further abstracts the published information providing privacy protection. Thus it seems reasonable to publish dependency collections without internal actions.

5.1 Simply Dependent Problems

The following defines simply dependent MA-Strips problems, where internal dependencies of public actions can be expressed by a dependency collection free of internal actions.

Definition 7. *An* MA-Strips *problem Π is* simply dependent *when there exists $\mathcal{D}$ such that* $\mathsf{actions}(\mathcal{D})$ *contains no internal actions and* $\mathcal{D} \simeq \mathsf{FD}(\Pi)$.

Suppose we have a simply dependent MA-Strips problem and a dependency collection $\mathcal{D}$ which proves the fact. In order to solve Π, once again, it is enough to solve only one local problem $\Pi \triangleright_{\mathcal{D}} \alpha$ (of an arbitrary agent α).

Lemma 2. *Let Π be a simply dependent* MA-Strips *problem. Let $\mathcal{D}$ be a dependency collection which proves that Π is simply dependent. Then $(\Pi \triangleright_{\mathcal{D}} \alpha) \simeq \Pi$ holds for any agent* $\alpha \in \mathsf{agents}(\Pi)$.

Proof. $(\Pi \triangleright_{\mathcal{D}} \alpha) \simeq (\Pi \triangleright_{\mathsf{FD}(\Pi)} \alpha) \simeq \Pi$.

The above method requires all the agents to publish the information from $\mathcal{D}$. However, the information does not need to be published to all the agents as it is enough to select one *trusted* agent and send the information only to him. Hence it is enough for all the agents to agree on a single trusted agent.

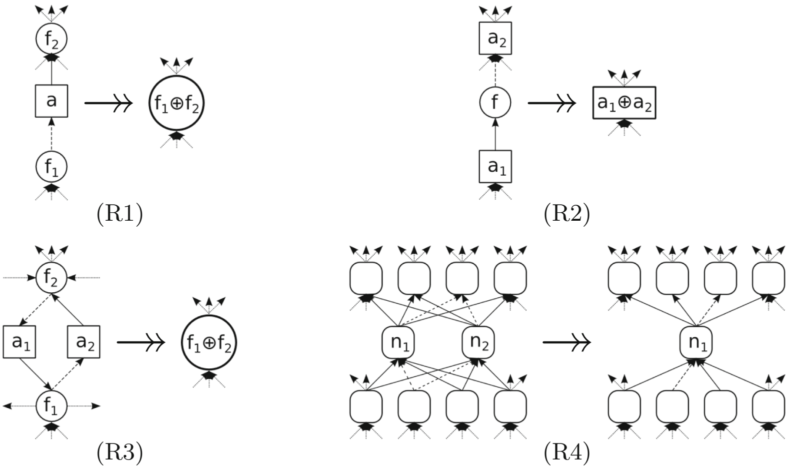
5.2 Dependency Graph Reductions

Recognizing simply dependent MA-Strips problems might be difficult in general. That is why we define an approximative method which can provably recognize some simply dependent problems. We define a set of reduction operations on dependency graphs and we prove that the operations preserve relation $\simeq$. Then we apply the reductions repeatedly starting with $\mathsf{FD}(\mathit{\Delta})$ obtaining a dependency graph which can not be reduced any further. This is done by every agent. When the resulting graphs contain no internal actions, then we know that the problem is simply dependent. Additionally, when the resulting graphs contain no internal facts, then we know that the problem is independent.

Our previous work [17] was restricted to problems where $\mathsf{pre}(a) = \mathsf{del}(a)$ holds for every action a. This impractical limitation is removed here. We still restrict our attention to problems where $\mathsf{del}(a) \subseteq \mathsf{pre}(a)$ holds for every action a. This is not considered limiting because a problem not meeting this requirement can be easily transformed to a permissible equivalent problem.

Finally, to abstract from the set of initial facts of a dependency graph $\mathit{\Delta}$, we introduce to the graph a special *initial action* $\langle \emptyset, \mathsf{init}(\mathit{\Delta}), \emptyset \rangle$. We suppose that every dependency graph has exactly one initial action and hence we do not need to remember the set of initial facts. The initial action is handled as *public* even when it has no public effect. Both definitions of dependency graphs are trivially equivalent but the one with an initial action simplifies the presentation of reduction operations.

We proceed by informal descriptions of dependency graph reductions. The formal definition is given below. The operations are depicted in Fig. 1.



(R1) (R2)

(R3) (R4)

Fig. 1. Graphical illustration of reduction operations (R1)–(R4). Circles represent fact nodes and rectangles represent action nodes. Rounded boxes in (R4) represent any node (either fact or action node).

- **(R1) Remove Simple Action Dependency.** If some internal action has only one delete effect and one add effect and there is no other action depending on f_1 we can merge both facts into one and remove that action.
- **(R2) Remove Simple Fact Dependency.** If some fact is the only effect of some action and there is only one action that consumes this effect without any side effects, we can remove this fact and merge both actions.
- **(R3) Remove Small Action Cycle.** In many domains, there are reversible internal actions that allow transitions between two (or more) states. All these states can be merged into a single state and the actions changing them can be omitted.
- **(R4) Merge Equivalent Nodes.** If two nodes (facts or actions) equal on incoming and outgoing edges, then we can merge these two nodes. Mostly this is not directly in the domain but this structure might appear when we simplify a dependency graph using the other reductions.
- **(R5) Remove Invariants.** After several reduction steps, it can happen that all the delete effects on some fact are removed and the fact is always fulfilled from the initial state. This happens, for example, in *Logistics*, where the location of a vehicle is internal knowledge and can be freely changed as described by reduction (R3). Once these cycles are removed, only one fact remains. The remaining fact represents that the vehicle is *somewhere*, which is always true. This fact can be freely removed from the dependency graph.

In order to formally define the above reductions we first define operator $[F]_{f_1 \to f_2}$ which renames fact f_1 to f_2 in the set of facts $F \subseteq P$.

$$[F]_{f_1 \to f_2} = \begin{cases} F & \text{if } f_1 \notin F \\ (F \setminus \{f_1\}) \cup \{f_2\} & \text{otherwise} \end{cases}$$

Similarly, we define operator $[F]_{\text{-}f} = F \setminus \{f\}$ which removes fact f from the set of facts F. These operators are extended to actions (applying the operator to preconditions, add, and delete effects) and to action sets (element-wise). The operators can be further extended to dependency graphs, where $[\Delta]_{\text{-}f}$ is the dependency graph determined by $[\mathsf{actions}(\Delta)]_{\text{-}f}$ and $[\mathsf{facts}(\Delta)]_{\text{-}f}$. Finally, for two actions a_1 and a_2 we define the merged action $a_1 \oplus a_2$ as the action obtained by unifying separately preconditions, add, and delete effects of both the actions.

The following formally defines *reduction relation* $\Delta_0 \to \Delta_1$ which holds when Δ_0 can be transformed to Δ_1 using one of the reduction operations.

Definition 8. *The* reduction relation $\Delta_0 \to \Delta_1$ *on dependency graphs is defined by the following four rules.*

(R1) *Rule (R1) is applicable to* Δ_0 *when*
 (1) Δ_0 *contains edges* $(f_1 \dashrightarrow a \to f_2)$,
 (2) a *is internal, and*
 (3) there are no other edges from/to a, *and*
 (4) there are no other edges from f_1.

Then $\varDelta_0 \rightarrow \varDelta_1$ *where* $\varDelta_1$ *is defined as* $\varDelta_1 = [\varDelta_0]_{f_1 \rightarrow f_2}$. *The initial action is preserved.*

(R2) *Rule (R2) is applicable to* $\varDelta_0$ *when*
(1) $\varDelta_0$ *contains edges* $(a_1 \rightarrow f \dashrightarrow a_2)$,
(2) there are no other edges from/to f, *and*
(3) there are no other edges from a_1, *and*
(4) a_2 *has no other delete effects, and*
(5) a_2 *is internal action.*
Then $\varDelta_0 \rightarrow \varDelta_1$ *where* $\varDelta_1$ *is given by the following.*

$$\begin{aligned}\mathsf{actions}(\varDelta_1) &= \{[a_1 \oplus a_2]_{\text{-}f}\} \cup (\mathsf{actions}(\varDelta_0) \setminus \{a_1, a_2\})\\ \mathsf{facts}(\varDelta_1) &= [\mathsf{facts}(\varDelta_0)]_{\text{-}f}\end{aligned}$$

If a_1 *is the initial action of* $\varDelta_0$ *then the new merged action becomes the initial action of* $\varDelta_1$. *Otherwise, the initial action is preserved.*

(R3) *Rule (R3) is applicable to* $\varDelta_0$ *when*
(1) $\varDelta_0$ *contains edges* $(f_1 \dashrightarrow a_1 \rightarrow f_2)$, *and*
(2) $\varDelta_0$ *contains edges* $(f_2 \dashrightarrow a_2 \rightarrow f_1)$, *and*
(3) a_1 *and* a_2 *are both internal, and*
(4) there are no other edges from/to a_1 *or* a_2.
Then $\varDelta_0 \rightarrow \varDelta_1$ *where* $\varDelta_1$ *is given by the following.*

$$\begin{aligned}\mathsf{actions}(\varDelta_1) &= [\mathsf{actions}(\varDelta_0) \setminus \{a_1, a_2\}]_{f_2 \rightarrow f_1}\\ \mathsf{facts}(\varDelta_1) &= [\mathsf{facts}(\varDelta_0)]_{f_2 \rightarrow f_1}\end{aligned}$$

The initial action is preserved as it is public.

(R4) *Rule (R4) is applicable to* $\varDelta_0$ *when* $\varDelta_0$ *contains two nodes* n_1 *and* n_2 *(either action or fact nodes) such that*
(1) nodes n_1 *and* n_2 *have equal sets of incoming and outgoing edges, and*
(2) n_1 *and* n_2 *are not public actions.*
Then $\varDelta_0 \rightarrow \varDelta_1$ *where, in the case* n_1 *and* n_2 *are actions,* $\varDelta_1$ *is given by the following.*

$$\begin{aligned}\mathsf{actions}(\varDelta_1) &= \{n_1 \oplus n_2\} \cup (\mathsf{actions}(\varDelta_0) \setminus \{n_1, n_2\})\\ \mathsf{facts}(\varDelta_1) &= \mathsf{facts}(\varDelta_0)\end{aligned}$$

When n_1 *or* n_2 *is the initial action of* $\varDelta_0$ *then the new merged action becomes the initial action of* $\varDelta_1$. *Otherwise, the initial action is preserved.*
In the case n_1 *and* n_2 *are facts,* $\varDelta_1 = [\varDelta_0]_{n_2 \rightarrow n_1}$.

(R5) *Let* a_{init} *be the initial action of* $\varDelta_0$. *Rule (R5) is applicable to* $\varDelta_0$ *when there exists fact* f *such that*
(1) $\varDelta_0$ *contains edge* $(a_{\mathit{init}} \rightarrow f)$, *and*
(2) $\varDelta_0$ *contains no edge* $(f \dashrightarrow a)$ *for any* a.
Then $\varDelta_0 \rightarrow \varDelta_1$ *where* $\varDelta_1$ *is defined as* $\varDelta_1 = [\varDelta_0]_{\text{-}f}$. *The initial action of* $\varDelta_1$ *is* $[a_{\mathit{init}}]_{\text{-}f}$.

The following defines *reduction equivalence relation* $\Delta_0 \sim \Delta_1$ as a reflexive, symmetric, and transitive closure of $\rightarrow$. In other words, Δ_0 and Δ_1 are *reduction equivalent* when one can be transformed to another using the reduction operations. Dependency collections $\mathcal{D}_0$ and $\mathcal{D}_1$ are *reduction equivalent* when graphs of corresponding agents are reduction equivalent.

Definition 9. *Dependency graphs* reduction equivalence relation*, denoted* $\Delta_0 \sim \Delta_1$*, is the least reflexive, symmetric, and transitive closure generated by the relation* $\rightarrow$*.*

Given MA-Strips *problem* Π*, dependency collections* $\mathcal{D}_0$ *and* $\mathcal{D}_1$ *of* Π *are* reduction equivalent*, written* $\mathcal{D}_0 \sim \mathcal{D}_1$*, when* $\mathcal{D}_0(\alpha) \sim \mathcal{D}_1(\alpha)$ *for any agent* $\alpha \in \mathsf{agents}(\Pi)$*.*

The following theorem formally states that reduction operations preserves public equivalence.

Theorem 2. *Let* Π *be an* MA-Strips *problem and let* $\mathsf{pre}(a) \subseteq \mathsf{del}(a)$ *hold for any internal action. Let* $\mathcal{D}_0$ *and* $\mathcal{D}_1$ *be dependency collections of problem* Π*. Then* $\mathcal{D}_0 \sim \mathcal{D}_1$ *implies* $\mathcal{D}_0 \simeq \mathcal{D}_1$*.*

Proof (Proof sketch). It can be shown that none of the reduction operations changes the set of public plans $\mathsf{sols}(\mathcal{D}_0(\alpha)) \triangleright \star$ of any agent $\alpha \in \mathsf{agents}(\Pi)$. Therefore repetitive application of reductions assures that $\mathcal{D}_0 \simeq \mathcal{D}_1$.

To avoid possible action confusion caused by value renaming, we suppose that actions are assigned unique ids which are preserved by the reduction, and that plans are sequences of these ids.

The consequences of the theorem are discussed in the following section.

5.3 Recognizing Simply Dependent Problems

Let us have an MA-Strips problem Π where $\mathsf{pre}(a) \subseteq \mathsf{del}(a)$ holds for every internal action a. Suppose that every agent α can reduce its full dependency collection $\mathsf{FD}(\alpha)$ to a state where it contains no internal action. Then there is $\mathcal{D}$ such that $\mathcal{D} \sim \mathsf{FD}(\Pi)$ and hence $\mathcal{D} \simeq \mathsf{FD}(\Pi)$ by Theorem 2. Hence Π is simply dependent and its public solution can be found without agent interaction, provided all the agents allow to publish $\mathcal{D}$. Important idea here is that publicly equivalent dependency graphs do not need to reveal the same amount of sensitive information. Moreover when $\mathcal{D} \sim \mathsf{MD}(\alpha)$ then Π is independent and can be solved without any interaction and without revealing other than public information. This gives us an algorithmic approach to recognize some independent and simply dependent problems.

6 Planning with Dependency Graphs

This section describes how agents use dependency graphs in order to solve MA-Strips problem Π (Algorithm 3). At first, every agent computes the dependency graph it is willing to share using function `ComputeSharedDG` described by Algorithm 2. Every agent α starts with the full dependency graph $\mathsf{FD}(\alpha)$ and tries to apply reduction operations repeatedly as long as it is possible. When the resulting reduced dependency graph $\varDelta_0$ contains only public actions, then the agent publishes $\varDelta_0$. Otherwise, the agent publishes only the minimal dependency graph $\mathsf{MD}(\alpha)$. Algorithm 2 clearly terminates for every input because every reduction decreases the number of nodes in the dependency graph. Hence the algorithm loop (lines 3–9 in Algorithm 2) can not be iterated more than n times when n is the count of nodes in $\mathsf{FD}(\alpha)$. Moreover, every reduction operation can be performed in a time polynomial to the size of the problem, and thus the whole algorithm is polynomial-time.

Algorithm 2. Compute the dependency graph to be published by agent α.

```
 1 Function ComputeSharedDG($\alpha$) is
 2     $\varDelta_0 \leftarrow \mathsf{FD}(\alpha)$;
 3     loop
 4         if $\exists \varDelta_1 : \varDelta_0 \rightarrow \varDelta_1$ then
 5             $\varDelta_0 \leftarrow \varDelta_1$;
 6         else
 7             break;
 8         end
 9     end
10     if $\varDelta_0$ contains only public actions then
11         return $\varDelta_0$;
12     else
13         return $\mathsf{MD}(\alpha)$;
14     end
15 end
```

Once the shared dependency graph $\varDelta$ is computed, Algorithm 3 continues by sending $\varDelta$ to other agents. Then shared dependency graphs of other agents are received. This allows every agent to complete the dependency collection $\mathcal{D}$, and to construct the local problem $\varPi \rhd_{\mathcal{D}} \alpha$. The rest of the planning procedure is the same as in the case of Algorithm 1.

The algorithm can be further simplified when all the agents succeeds in reducing $\mathsf{FD}(\alpha)$ to an equivalent dependency graph without internal actions, that is, when $\varPi$ is provably simply dependent. Then it is enough to select one agent to compute public solution of $\varPi$. When at least one agent α fails to share dependency collection equivalent to the full dependency collection $\mathsf{FD}(\alpha)$ then iterated negotiation is required. When some agent α (but not all the agents) succeeds in reducing $\mathsf{FD}(\alpha)$ then every plan created by any other agent will be automatically α-extensible.

Algorithm 3. Distributed planning with dependency graphs.

```
Function DgPlanDistributed($\alpha$) is
    $\mathit{\Delta} \leftarrow$ ComputeSharedDG($\alpha$);
    send $\mathit{\Delta}$ to other agents;
    construct $\mathcal{D}$ from other agent's graphs;
    compute local problem $\mathit{\Pi} \triangleright_{\mathcal{D}} \alpha$;
    return MaPlanDistributed($\mathit{\Pi} \triangleright_{\mathcal{D}} \alpha$);
end
```

7 Privacy Leakage Analysis

Although distributed planning with the dependency graphs trades exposition of (reduced) private information for efficiency, from perspective of the worst case, there is no extra private knowledge shared. In this section, we will support this claim by analysis of what information is leaked by sharing the reduced dependency graph $\mathit{\Delta}$.

According to [3], a multiagent planning algorithm is *strongly privacy preserving* if no agent can deduce any information about private facts and private preconditions/effects of any action, beyond what can be deduced from the public projection of the planning problem and the public projection of the solution plan.

The shared reduced dependency graph $\mathit{\Delta}$ can be, in some cases, equivalent to the original dependency graph. In these cases, *renamed* internal facts are shared between the agents. Nevertheless other agents do not know, whether any reduction has been performed, and so they see a dependency graph which can represent an unlimited number of different dependency graphs with an *equivalent reduction*. The definition of strong privacy allows an agent to share any knowledge which can be deduced from own actions description, the public projection of other agents actions, and the public projection of the solution plan. Not much information can be deduced from the solution plan when solving a single problem. In fact, other agents can deduce only the information that given sequence of actions is possible and thus no action deletes preconditions of the immediately following action.

Suppose a situation, when we observe an agent α using a strongly privacy preserving algorithm in long term. The most extreme case is when we know all possible plans in which the agent can participate in all different problems and we also know that all the problems are unsolvable for this agent, thus which public plans are not α-extensible. We can suppose that all these plans contain all agent's published actions and all their possible combinations. From this knowledge we could deduce what dependencies are between public actions appearing in these plans. Certainly, this deduced information will contain all the information contained in the shared dependency graph $\mathit{\Delta}$. This gives us the first insight about the amount of information which is contained in the shared dependency graph. We can state that it contains information which could be eventually deduced if we follow the agent planning of all possible different planning tasks.

Let us now compare sharing of dependency graph with privacy which an agent leaks during a single run of Psm-vr in the worst case. Suppose that we have two agents α and β with Psm-vr algorithm. During one iteration, both agents update their Φ and share it with the other agent to check whether there is a non-empty intersection. In the worst case, it can happen, that the agent α publishes all its possible plans, therefore Φ_α is complete and no longer changing, while the agent β still updates his Φ_β. Then, agent β can reconstruct all the agent's α dependencies between its public actions. Again, shared dependency graph Δ is certainly a subset of the knowledge β would deduce in this case.

Even though agents are not able to detect the presence of the internal fact from a single execution of the Psm-vr algorithm, the agents can deduce the existence of this fact when cooperating longer time or when it takes long time before the agents find a common solution. For example, suppose that there is no solution at all. In that case, each agent publishes all his possible local solution and then announces that there are no more local solutions. Then, similarly to the previous case, each other agent can see whether some action has to always precede another action. If such case exists, the agent deduce that there is a fact or some other dependency between these actions.

Example 2 (Logistics with three cities). Let's suppose following *logistic* problem with tree cities (*cityA, cityB* and *cityC*) and two agents (*Truck* transporting the *package* between cities *cityA* and *cityB*, and *Plane* transporting the *package* between cities *cityB* and *cityC*), where the locations of agents and their load are private and the package location is public only when it is present in *cityB* where the agents transship the *package*.

Agent *Truck* starts to sequentially generate following possible solutions – we show only *Truck*'s own actions and omit the actions planned for *Plane* agent, because only these actions can reveal some knowledge about its internal facts and actions.

- $\langle\rangle$
- $\langle unload(package, cityB)\rangle$
- $\langle unload(package, cityB), load(package, cityB)\rangle$
- $\langle unload(package, cityB), load(package, cityB), unload(package, cityB)\rangle$
- ...
- $\langle \Big(unload(package, cityB), load(package, cityB) \Big)^* \rangle$
- $\langle \Big(unload(package, cityB), load(package, cityB) \Big)^*, unload(package, cityB)\rangle$

Even though there is an infinite number of different public plans, they can be compactly represented in finite structure (for example Psm-vr uses *Planning state machines* – see Sect. 8 for more details).

From these plans we can observe:

- all plans start with $unload(package, cityB)$ action, and
- a $load(package, cityB)$ action has to be between two $unload(package, cityB)$ actions.

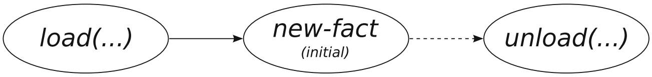


Fig. 2. Example of dependency graph containing all knowledge which can leak during *logistics* planning.

Moreover, we can see that there has to be *unload(package, cityB)* before *load(package, cityB)*, but these actions are already dependent through a public fact *inLocation(package, Prague)* and thus it does not imply any internal connection. Similar situation is for the observation that between two *load(package, cityB)* actions has to be *unload(package, cityB)* action.

These observations can be modeled by a dependency graph (Fig. 2) containing one internal fact that is also present in the initial state. This fact is consumed by $unload(package, cityB)$ and produced by $load(package, cityB)$.

We can conclude that the Psm-vrd algorithm does not publish any more internal information that the Psm-vr algorithm in the worst case. In fact, the most significant difference between these two algorithms is that Psm-vrd exposes exactly the internal information that could be exposed by Psm-vr in the worst case and which can be described in the form of simple dependencies between the public actions.

8 PSM Planning System

In this section we describe the implementation of presented planners Psm-vr and Psm-vrd in details. Algorithm 4 shows the full listing including several optional blocks that allow to significantly improve the performance of the planner. All planner configurations are sound and complete. We firstly describe the basic version of the planner and then, in Sect. 8.1 we describe the optional improvements. Psm-vrd contains all optional parts while Psm-vr contains all but sharing of the dependency graphs.

The sets of plans Φ_α are represented using *planning state machines*, PSMs, which are based on common finite state machines. PSMs allows to effectively represent large sets of plans and also to effectively implement the intersection of different agents sets of plans. Consult [19] for more details on algorithms and implementation.

The planners' input problems are supposed to be provided in a PDDL files which might contain parametric actions. Our algorithm requires grounded actions and hence the input is grounded as the first step. Specific implementation of method `GroundProblem` is crucial. It has to generate all possible usable action instances but it should not generate any action instance which is never applicable. Otherwise, other agent could use that action in their plans. Agents collaboratively create distributed planning graph and then ground their problem to actions that appear in this planning graph. This approach allows to protect agents privacy.

Algorithm 4. Distributed MA planning algorithm.

```
Function MaPlanDistributed($\Pi \triangleright \alpha$) is
    $\Pi_\alpha \leftarrow$ GroundProblem($\Pi \triangleright \alpha$);
    // (Optional) Compute and exchange dependency graphs
    $\Pi_\alpha \leftarrow$ ComputeDependencies($\Pi_\alpha$);
    if all agents published dependency graphs then
        return any solution of $\Pi_\alpha \triangleright \star$;
    end
    // (Optional) Compute relaxed solution landmarks
    $\pi_R \leftarrow$ CreateRelaxedPlan($\Pi$);
    $\Pi_\alpha \leftarrow$ AddLandmarks($\Pi_\alpha, \pi_R \triangleright \alpha$);
    $\Phi_\alpha \leftarrow \emptyset$;
    loop
        generate new $\pi_\alpha \in \mathsf{sols}(\Pi \triangleright \alpha)$;
        // (Optional) Verify created plan
        $\pi_\alpha \leftarrow$ VerifyPlan($\pi_\alpha \triangleright \star$);
        $\Phi_\alpha \leftarrow \Phi_\alpha \cup \{\pi_\alpha \triangleright \star\}$;
        exchange public plans $\Phi_\beta$ with other agents;
        $\Phi \leftarrow \bigcap_{\beta \in \mathsf{agents}(\Pi)} \Phi_\beta$;
        if $\Phi \neq \emptyset$ then
            return $\Phi$;
        end
        // (Optional) Use received knowledge in future planning
        forall the $\beta \in \mathsf{agents}(\Pi)$ do
            $\Pi_\alpha \leftarrow$ AddLandmarks($\Pi_\alpha, \Phi_\beta$);
        end
    end
end
```

Every agent keeps generating new plans from $\mathsf{sols}(\Pi \triangleright \alpha)$ and shares their public projections with other agents. Local planning problems are common STRIPS problems which can be solved by any classical STRIPS planner. The only special requirement on the underlying planner is that the planner must be able to generate a plan which differs from the previously generated plans. In our implementation, we use a modified version of FastDownward[2] planner [11].

Our modification allows the planner to generate a plan which differs from plans provided as an input. This extension is inspired by diverse planning with homotopy class constraints [2]. Homotopy classes of plans are naturally defined by their public projections, that is, two plans belong to same homotopy class iff they have equal public projection.

The algorithm ends with a set of public solutions Φ which contains possibly several extensible public plans. One random public plan $\sigma \in \Phi$ is selected for the

---

[2] http://www.fast-downward.org/.

reconstruction of a solution of the original problem Π. For that, each agent α creates a classical Strips *reconstruction* problem Π_α which contains only internal actions of agent α. To problem Π_α, the generated public solution σ is added as *hard action landmarks* to ensure that all the landmark actions are executed. The public plan is α-extensible and thus problem Π_α is solvable. Agents then merge together solutions of all the reconstruction problems and create a solution of the original problem Π.

8.1 Optional Extensions

The basic algorithm, where agents iteratively generate new plans, exchange their public projections and check whether their intersection is nonempty, can be extended to achieve better results. In this article we focused on publishing internal dependencies between public action, which allows agents to exchange some abstract rules describing their internal knowledge. In this section we briefly describe three more implemented extensions. First extension, described in Sect. 8.2, significantly reduces number of misleading landmarks by use of plan verification. Section 8.3 describes how the generation of new plans is driven towards the non-empty intersection of Φs. In Sect. 8.4, we describe the second extension, which computes a relaxed solution π and uses its public projection $\pi \triangleright \star$ as initial landmarks.

8.2 Plan Verification and Analysis

Psm-vr planner uses public plans generated by other agents as landmarks to guide future plan search (Algorithm 4 line 12). However, it is desirable to use only extensible plans to guide plan search because non-extensible plans cannot lead to a non-empty intersection Φ. Every generated plan should be verified by other agents in order to determine its extensibility. However, extensibility (or α-extensibility) checking is expensive and thus we propose only an approximative method of plan verification.

In order to approximative α-extensibility we use generic process calculi type system scheme Poly✶ [12,15] to determine the least provably unreachable action in a public plan. The result of Poly✶ analysis is either indeterminate, or the index an action which is guaranteed unreachable. However, some action prior to the indexed action might be actually unreachable as well because Poly✶ analysis provides only a polynomial time approximation of α-extensibility.

Described α-extensibility approximation suggests the following implementation of method `VerifyPlan(`$\pi_\alpha \triangleright \star$`)`. When agent α generates a new plan π_α, it sends its public projection $\pi_\alpha \triangleright \star$ to all the other agents. Once other agent β receives $\pi_\alpha \triangleright \star$, it runs the above β-extensibility check on it, and sends the result back to agent α. Agent α collects analysis results from all the other agents and strips the plan π to prefix acceptable for all agents. Finally only the stripped plan is used as a landmark to guide future plan search. We can even further speed up convergence of the Psm-vr by forbidding the generation of plans with public prefixes matching plans already refused by other agents. See [11] for more detailed description.

8.3 Heuristic Plan Generation

In Psm-vr the generation of new local plans is guided by the knowledge obtained from Φ_β's of other agents so that the plans similar to those received by other agents are preferred. The knowledge from Φ_β's of other agents is compiled into the local problem by function `AddLandmarks` using the principle of *prioritizing actions* and *soft-landmarks*. Prioritizing actions are implemented using action costs so that internal actions are preferred to public actions, and α's public actions are preferred to other agent actions. When agent α finds a new local solution it sends its public projection to all the other agents. Other agents then extend their local problem $\Pi \triangleright \beta$ to contain duplicated landmark actions from the received plan. These landmark actions have significantly decreased cost and they are interlinked using additional facts to ensure they are used in the order suggested by the public plan.

8.4 Initial Relaxed Plan Landmarks

The delete effect relaxation, where delete effects of actions are ignored, has proved its relevance both in Strips planning [10], and recently also in MA-Strips planning [22]. It is known that to find a solution of a relaxed problem is an easier task than to find a solution of the original problem.

Our algorithm first creates solution π_R of the relaxed problem and then transforms it into *initial landmarks*, which are used by all the agents. When $\pi_R \triangleright \star$ is extensible then every agent α is likely to generate local solution π_α such that $\pi_\alpha \triangleright \star = \pi_R \triangleright \star$ in the first iteration. In that case the algorithm terminates directly in the first iteration causing a dramatic speed-up. Otherwise, the initial landmark is forgotten by all the agents and the algorithm continues by the second iteration as before.

When creating the relaxed plan π_R, the agents compute distributed planning graph by trying to fulfill goals or preconditions of reachable actions of other agents. In the case of domains where actions model use of *limited resources*, it is better to find solutions where each agent fulfill few of the goals, because limiting resources are relaxed away. Therefore, during the extraction of the relaxed plan from the distributed planning graph, agents are lazy and try to fulfill only one goal and they pass the remaining goals together with new subgoals to another agent. More effective implementation would make use of *exploration queues* [22].

9 Experiments

For experimental evaluation we use benchmark problems from the CoDMAP'15 competition[3]. The benchmark set contains 12 domains with 20 problems per

---

[3] See http://agents.fel.cvut.cz/codmap.

domain. Each agent has its own domain and problem files containing description of known facts and actions. Additionally, some facts/predicates are specified as private and thus should not be communicated to other agents. The privacy classification roughly corresponds to MA-Strips.

We firstly present analysis of internal dependencies and their reductions in Sect. 9.1. In Sect. 9.2, we present results independently evaluated by organizers of the CoDMAP'15 competition.

9.1 Domain Analysis

In this section we present analysis of internal dependencies of public action in the benchmark problems.

We have evaluated internal dependencies of public actions within benchmark problems by constructing full dependency graph for every agent in every benchmark problem. We have applied Algorithm 2 to reduce full dependency graphs to an irreducible publicly equivalent dependency graph. The results of the analysis are presented in Table 1. The table columns have the following meaning. Column **(Facts)** represents an average number of all facts in a domain problem. Column **(Public facts)** represents an average number of public facts in a domain problem. Column **(Merge facts)** represents an average size of $\mathsf{facts}(\varDelta)$ in the resulting irreducible dependency graph. Column **(Fact disclosure)** represents the percentage of published merge facts with respect to all the internal facts. Column **(Actions)** represents an average number of all actions in a domain problem. Column **(Public actions)** represents an average number of public actions in a domain problem. Column **(Success)** represents the percentage of agents capable of reducing their full dependency graph to a publicly equivalent graph without internal actions.

We can see that five of the benchmark domains, namely *Blocksworld*, *Depots*, *Driverlog*, *Logistics*, *Taxi*, *Wireless*, and *Woodworking* were found simply dependent. All the problems in these domains can be solved by solving a local problem

Table 1. Results of the analysis of internal dependencies of public actions in benchmark domains.

Domain	Facts	Public facts	Merge facts	Fact disclosure	Actions	Public actions	Success
Blocksworld	787	733	53	100%	1368	1368	100%
Depots	1203	1139	56	85%	2007	2007	100%
Driverlog	1532	1419	16	25%	7682	7426	100%
Elevators	509	343	43	29%	2060	1767	70%
Logitics	240	154	56	63%	342	298	100%
Rovers	2113	1251	31	3%	3662	1555	13%
Satellite	846	578	0	0%	8839	914	1%
Taxi	177	173	0	0%	107	107	100%
Woodworing	1448	1425	7	27%	4126	4126	100%
Zenotravel	1349	1204	0	0%	13516	2364	0%

of a single agent. On the contrary, in most problems of domains *Rovers*, *Satellite*, and *Zenotravel*, none of the agents were able to reduce its full dependency graph so that it contains no internal actions. Hence the agents in these domain publish only the minimal dependency graphs and hence the analysis does not help in solving them. Finally, in *Elevators* domain, some of the agents succeeded in reducing their full dependency graphs and thus the analysis can partially help to solve them.

9.2 Experimental Results

To evaluate the impact of dependency analysis, we use our PSM-based planners [19] submitted to the CoDMAP'15 competition. Namely, we use planner PSM-VR [20] and its extension with dependency analysis PSM-VRD.

Figure 3 evaluates the impact of the dependency analysis on CoDMAP benchmark problems. For each problem, a point is drawn at the position corresponding to the runtime without dependency analysis (x-coordinate) and the runtime with dependency analysis (y-coordinate). Hence the points below the diagonal constitute improvements. Results show that the dependency analysis decreases overall planning time of PSM algorithm. We can see that in few cases the time increases which is caused by the time consumed by reduction process. Also, by publishing additional facts, the problem size can grow and thus it can become harder to solve.

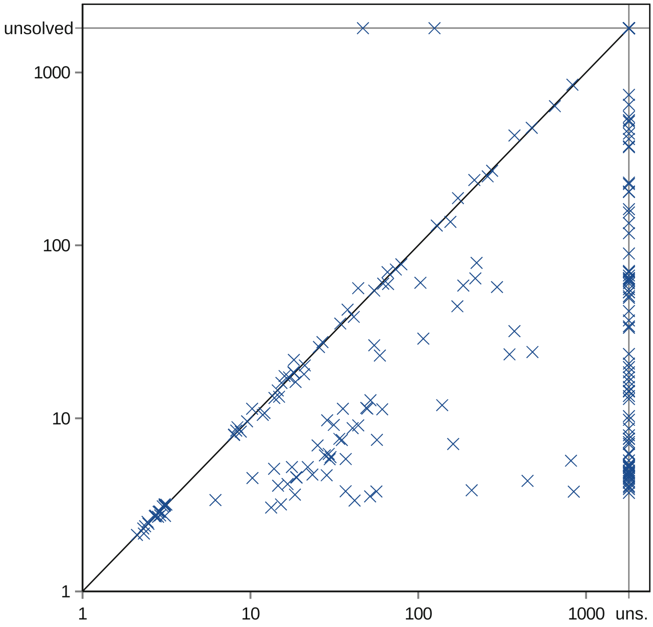


Fig. 3. Comparison of planning times (in seconds) of PSM-VR algorithm without (X axis) and with (Y axis) internal problem reduction.

Table 2. Results at CoDMAP competition (http://agents.fel.cvut.cz/codmap/results/). Top table shows overall coverage of solved problem instances. Bottom table shows IPC score over the plan quality Q (a sum of Q^*/Q over all problems, where Q^* is the cost of an optimal plan, or of the best plan found by any of the planners for the given problem during the competition).

Domain	#	MAPlan LM-Cut[a]	MAPlan MA-LM-Cut[a]	MH-FMAP	MAPlan FF+DTG	PSM-VR	PSM-VRD
Blocksworld	20	2	1	0	14	12	**20**
Depots	20	5	2	2	10	1	**16**
Driverlog	20	15	9	18	18	16	**20**
Elevators	20	2	0	**9**	**9**	2	5
Logitics	20	4	5	4	**16**	0	**16**
Openstacks	20	1	1	8	**18**	14	**18**
Rovers	20	2	4	18	**19**	13	13
Satellite	20	13	4	4	14	7	**17**
Taxi	20	19	14	**20**	19	9	**20**
Wireless	20	3	2	0	**4**	0[b]	0[b]
Woodworing	20	3	4	8	14	9	**19**
Zenotravel	20	6	6	16	**19**	16	16
Coverage	240	75	52	107	174	99	**180**
Domain	#	MAPlan LM-Cut[a]	MAPlan MA-LM-Cut[a]	MH-FMAP	MAPlan FF+DTG	PSM-VR	PSM-VRD
Blocksworld	20	2	1	0	7	11	**17**
Depots	20	5	2	2	6	1	**15**
Driverlog	20	15	9	**17**	12	14	16
Elevators	20	2	0	**8**	6	1	4
Logitics	20	4	5	4	13	0	**15**
Openstacks	20	1	1	8	**18**	9	12
Rovers	20	2	4	**18**	16	5	5
Satellite	20	**13**	4	4	10	6	**13**
Taxi	20	**19**	14	17	15	6	16
Wireless	20	3	2	0	**4**	0[b]	0[b]
Woodworing	20	3	4	7	13	8	**17**
Zenotravel	20	6	6	**15**	**15**	10	10
IPC Score	240	75	52	100	135	72	**140**

[a] This is optimal planner. [b] This post-submission domain was not supported by the planner parser. These results are presented with the consent of CoDMAP organizers.

Tables 2 and 3 show official results of the CoDMAP competition. We can see that the dependency analysis significantly improved the performance of PSM-VR planner. Moreover, PSM-VRD achieved the overall best coverage in 8 out of 12 domains. As expected, the highest coverage directly corresponds to the success

Table 3. Results at CoDMAP competition. Table shows IPC Agile score over the planning time T (a sum of $1/(1 + \log_{10}(T/T^*))$ over all problems, where T^* is the runtime of the fastest planner for the given problem during the competition).

Domain	#	MAPlan LM-Cut[a]	MAPlan MA-LM-Cut[a]	MH-FMAP	MAPlan FF+DTG	PSM-VR	PSM-VRD
Blocksworld	20	1	0	0	**14**	5	**14**
Depots	20	3	1	1	9	0	**14**
Driverlog	20	10	4	11	**17**	7	14
Elevators	20	1	0	4	**8**	1	4
Logitics	20	3	2	2	13	0	**14**
Openstacks	20	0	0	3	**18**	7	8
Rovers	20	1	1	7	**19**	6	6
Satellite	20	10	2	1	**13**	3	12
Taxi	20	14	7	10	**19**	3	15
Wireless	20	**3**	2	0	2	0[b]	0[b]
Woodworing	20	2	3	4	9	5	**18**
Zenotravel	20	5	4	10	**18**	8	8
Agile Score	240	52	27	52	**159**	45	127

[a] This is optimal planner. [b] This post-submission domain was not supported by the planner parser. These results are presented with the consent of CoDMAP organizers.

Table 4. Results at CoDMAP competition. Table shows sum of times (in seconds) needed to solve selected problems. Selected problems contain problems solved by all the planners.

Domain	#	MAPlan LM-Cut[a]	MAPlan MA-LM-Cut[a]	MH-FMAP	MAPlan FF+DTG	PSM-VR	PSM-VRD
Depots	1	2	3	12	**1**	56	4
Driverlog	9	18	622	28	**14**	172	39
Rovers	1	1296	1396	47	**1**	15	13
Satellite	2	1634	496	17	**2**	22	22
Taxi	7	18	331	212	**11**	2525	32
Woodworing	1	4	5	25	**2**	29	6
Zenotravel	6	**11**	57	23	1806	54	53
Total	27	2983	2910	363	1836	2873	**168**

[a] This is optimal planner. These results are presented with the consent of CoDMAP organizers.

of dependency analysis. The table also shows results of two additional criteria comparing the quality (IPC Score) of solutions and the time (IPC Agile Score) need to find the solution. In both criteria PSM-VRD performed very well even though it was outperformed by MAPlan-FF+DTG planner in the IPC Agile Score.

Table 4 shows times the planners needed to solve problems. Problems contain only those solved by all the planners. Some planners did not solved any problems of some domains and thus these domains were omitted. We can see that even though PSM-VRD did not solved any problem in the fastest time, it outperformed other planners in total. Reductions of internal dependencies allowed achieve more uniform planning times (especially, it significantly reduced long planning times of *Driverlog* and *Taxi* domains) and in total to improve the performance of PSM-VR more than 17-times.

10 Conclusions

Motivation for our work was to improve efficiency of coordination-based multiagent planning in deterministic environment by extracting and utilizing additional information on dependencies among agent's public actions.

First, we have formally and semantically defined internally independent and simply dependent MA-STRIPS problems and proposed a set of reduction rules utilizing the underlying dependency graph. To identify and extract internally independent and simply dependent problems, we have proposed technique which can build a full dependency graph and try to reduce it to an irreducible publicly equivalent dependency graph. This principle provides an algorithmic procedure for recognizing provably internally independent and simply dependent problems, which can be solved without agent interaction. The proposed reduction rules were defined over structural information of the dependency graph and provided recursive removal of superfluous facts and actions by analysis of simple dependency, cycles, equivalency, and state invariants.

Second, we have provided an analysis of impacts of usage of the dependency graphs on privacy preservation in multiagent planning. We have shown that the principle does not leak any additional information in contrast to worst-case for planning without the action dependency information.

Finally, we have summarized implementation details of the PSM planner and extended its current variant PSM-VR utilizing plan verification and relaxed planning heuristics towards PSM-VRD including the reduction of action dependencies. We showed that reduction of the standard multiagent planning benchmarks using the dependencies provides overall 71% downsizing, more than 17× acceleration and nearly doubled the number of solved problems by PSM-VRD in comparison to PSM-VR. In comparison with the latest distributed multiagent planners, PSM-VRD outperformed all and won the distributed track of the recent multiagent planning competition CoDMAP.

Acknowledgements. This research was supported by the Czech Science Foundation (no. 15-20433Y) and by the Czech Ministry of Education (no. SGS13/211/OHK3/3T/13). Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the program "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.

References

1. Bäckström, C., Jonsson, A., Jonsson, P.: Macros, reactive plans and compact representations. In: ECAI 2012, pp. 85–90 (2012). https://doi.org/10.3233/978-1-61499-098-7-85

2. Bhattacharya, S., Kumar, V., Likhachev, M.: Search-based path planning with homotopy class constraints. In: Felner, A., Sturtevant, N.R. (eds.) SOCS. AAAI Press (2010). http://dblp.uni-trier.de/db/conf/socs/socs2010.html#BhattacharyaKL10
3. Brafman, R.I.: A privacy preserving algorithm for multi-agent planning and search. In: Yang, Q., Wooldridge, M. (eds.) Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, 25–31 July 2015. pp. 1530–1536. AAAI Press (2015). http://ijcai.org/Abstract/15/219
4. Brafman, R.I., Domshlak, C.: From one to many: planning for loosely coupled multi-agent systems. In: ICAPS 2008, pp. 28–35 (2008)
5. Chen, Y., Yao, G.: Completeness and optimality preserving reduction for planning. In: Proceedings of 21st IJCAI, pp. 1659–1664 (2009)
6. Chrpa, L.: Generation of macro-operators via investigation of action dependencies in plans. Knowl. Eng. Rev. **25**(3), 281–297 (2010). https://doi.org/10.1017/S0269888910000159
7. Coles, A., Coles, A.: Completeness-preserving pruning for optimal planning. In: Proceedings of 19th ECAI, pp. 965–966 (2010)
8. Fikes, R., Nilsson, N.: STRIPS: a new approach to the application of theorem proving to problem solving. In: IJCAI 1971, pp. 608–620 (1971)
9. Haslum, P.: Reducing accidental complexity in planning problems. In: Proceedings of 20th IJCAI, pp. 1898–1903 (2007)
10. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. J. Artif. Intell. Res. (JAIR) **14**, 253–302 (2001). https://doi.org/10.1613/jair.855
11. Jakubův, J., Tožička, J., Komenda, A.: Multiagent planning by plan set intersection and plan verification. In: Proceedings of ICAART 2015 (2015)
12. Jakubův, J., Wells, J.B.: Expressiveness of generic process shape types. In: Wirsing, M., Hofmann, M., Rauschmayer, A. (eds.) TGC 2010. LNCS, vol. 6084, pp. 103–119. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15640-3_8
13. Jonsson, A.: The role of macros in tractable planning. J. Artif. Intell. Res. **36**, 471–511 (2009)
14. Jonsson, P., Bäckström, C.: Tractable plan existence does not imply tractable plan generation. Ann. Math. Artif. Intell. **22**, 281–296 (1998)
15. Makholm, H., Wells, J.B.: Instant polymorphic type systems for mobile process calculi: just add reduction rules and close. In: Sagiv, M. (ed.) ESOP 2005. LNCS, vol. 3444, pp. 389–407. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31987-0_27
16. Nissim, R., Brafman, R.I.: Multi-agent A* for parallel and distributed systems. In: Proceedings of AAMAS 2012, pp. 1265–1266 (2012)
17. Tožička, J., Jakubův, J., Komenda, A.: On internally dependent public actions in multiagent planning. In: Proceedings of DMAP Workshop of ICAPS 2015 (2015)
18. Tožička, J., Jakubův, J., Durkota, K., Komenda, A., Pěchouček, M.: Multiagent planning supported by plan diversity metrics and landmark actions. In: Proceedings of ICAART 2014 (2014)
19. Tožička, J., Jakubův, J., Komenda, A.: Generating multi-agent plans by distributed intersection of finite state machines. In: ECAI 2014, pp. 1111–1112 (2014)
20. Tožička, J., Jakubův, J., Komenda, A.: PSM-based planners description for CoDMAP 2015 competition. In: CoDMAP 2015 (2015)

21. Tožička, J., Jakubuv, J., Komenda, A., Pěchouček, M.: Privacy-concerned multiagent planning. Knowl. Inf. Syst. **48**, 581–618 (2015). http://link.springer.com/10.1007/s10115-015-0887-7
22. Štolba, M., Komenda, A.: Relaxation heuristics for multiagent planning. In: 24th International Conference on Automated Planning and Scheduling (ICAPS), pp. 298–306 (2014)

Controlling a Single Transport Robot in a Flexible Job Shop Environment by Hybrid Metaheuristics

Houssem Eddine Nouri[1,2](✉), Olfa Belkahla Driss[1,3], and Khaled Ghédira[1,2]

[1] SOIE-COSMOS, École Nationale des Sciences de l'Informatique,
Université de la Manouba, Tunis, Tunisia
`houssemeddine.nouri@gmail.com`, `olfa.belkahla@isg.rnu.tn`,
`khaled.ghedira@anpr.tn`
[2] Institut Supérieur de Gestion de Tunis, Université de Tunis,
Bardo, Tunis, Tunisia
[3] École Supérieure de Commerce de Tunis, Université de la Manouba,
Tunis, Tunisia

Abstract. In robotic systems, the control of some elements such as transport robot has some difficulties when planning operations dynamically. The Flexible Job Shop scheduling Problem with Transportation times and a Single Robot (FJSPT-SR) is a generalization of the classical Job Shop scheduling Problem (JSP) where a set of jobs additionally have to be transported between machines by a single transport robot. Hence, the FJSPT-SR is more computationally difficult than the JSP presenting two NP-hard problems simultaneously: the flexible job shop scheduling problem and the robot routing problem. This paper proposes a hybrid metaheuristic approach based on clustered holonic multiagent model for the FJSPT-SR. Firstly, a scheduler agent applies a Neighborhood-based Genetic Algorithm (NGA) for a global exploration of the search space. Secondly, a set of cluster agents uses a tabu search technique to guide the research in promising regions. Computational results are presented using benchmark data instances from the literature of FJSPT-SR. New upper bounds are found, showing the effectiveness of the presented approach.

Keywords: Scheduling · Transport · Robot · Genetic algorithm
Tabu search · Holonic multiagent

1 Introduction

Scheduling is a field of investigation which has known a significant growth these last years. The scheduling problems appear in all the economic areas, from computer engineering to industrial production and manufacturing. The Job Shop scheduling Problem (JSP), which is among the hardest combinatorial optimization problems [34], is a branch of the industrial production scheduling problems. The JSP is known as one of the most popular research topics in the literature

© Springer International Publishing AG, part of Springer Nature 2018
N. T. Nguyen et al. (Eds.): TCCI XXVIII, LNCS 10780, pp. 93–115, 2018.
https://doi.org/10.1007/978-3-319-78301-7_5

due to its potential to dramatically decrease costs and increase throughput [20]. The Flexible Job Shop scheduling Problem with Transportation times and a Single Robot (FJSPT-SR) is a generalization of the classical JSP where a set of jobs additionally have to be transported between machines by a single transport robot. Hence, the FJSPT-SR is more computationally difficult than the JSP presenting an additional difficulty caused by a set of jobs to be transported by a single robot between a set of available machines. In the FJSPT-SR, we have to consider two NP-hard problems simultaneously: the flexible job shop scheduling problem [26,30] and the robot routing problem, which is similar to the pickup and delivery problem [27].

For the literature of the Job Shop scheduling Problem with Transportation times and a Single Robot, most of the researchers have considered the machine and robot scheduling as two independent problems. Therefore, only few researchers have emphasized the importance of simultaneous scheduling of jobs and the single robot.

To solve this problem, mathematical formulations are used to find optimal solutions for this problem, but the complexity of some large instances allowed to increase the processing time for some important solutions. [32] proposed a mixed integer programming formulation for this problem, and they assumed that the robot always returns to the load/unload station after transferring a load, which reduces the flexibility of the robot and influences the overall schedule length. An integer programming model is formulated by [5] for the machine and robot scheduling problems with a set of time window constraints. According to the authors, the resulting model is intractable in practice, because of its nonlinearity and its size. A linear programming formulation is modeled by [17,18] for a simultaneous scheduling of machines and one single robot generating the optimal lower bound results for this problem. [10] adapted a mathematical formulation for a shop scheduling problem with one transporter robot. This formulation differed from the published works because it considered the maximum number of jobs authorized in the system, the upstream and downstream storage capacities and the robot loaded/unloaded movements.

Moreover, heuristic and metaheuristic methods offered new opportunity to find solutions for this problem in a reasonable time but they did not guaranteed the optimality. [31] implemented a Branch and Bound procedure for a simultaneous scheduling of machines and resources handling in a Job shop environment. But, they did not take into consideration the violation of precedence relations between the different machine operations belonging to the same job. An iterative heuristic is used by [5] based on the decomposition of the master problem into two sub-problems, allowing a simultaneous resolution of this scheduling problem with time windows. [36] adapted a genetic algorithm for this scheduling problem in a flexible manufacturing system, and where they used a chromosome representation composed of two parts, the operation task sequencing and the transport resource assignment. [2] treated the simultaneous machine and robot scheduling problem using a forward propagation heuristic, and where they supposed that the robot movements between cells are considered as additional machines. A local search algorithm is proposed in [17,18] for the job shop scheduling problem with a single robot, where they supposed that the robot movements can be considered as a generalization of the travelling salesman problem with time windows, and additional precedence constraints must be respected. The used local search is based on a neighborhood structure inspired from [28] to make the search process more effective. [1] addressed the problem of simultaneous scheduling of machines and identical robots in flexible manufacturing systems, by developing a hybrid approach composed of a genetic algorithm and a heuristic. The genetic algorithm is used for the jobs scheduling problem and the robot assignment is made by the heuristic algorithm. A hybrid multi-objective genetic algorithm is proposed by [33] to solve this combined problem, and considered three minimization objectives, which are the makespan, mean flow time and mean tardiness. [23] studied the job shop scheduling problem with several transport robots, where they used a local search algorithm based on a neighbourhood generated by permutation of two operations or by assigning another robot to a transport operation. [11] addressed the simultaneous scheduling problem of machines and robots in flexible manufacturing systems, by proposing new solution representation based on robots rather than machines. Each solution is evaluated using a discrete event approach. An efficient neighbouring system is then implemented into three different metaheuristics: iterated local search, simulated annealing and their hybridisation. A differential evolution algorithm is developed by [3] for the machines and two robots scheduling problem, this algorithm is inspired by [35] which was proposed for the Chebyshev Polynomial Fitting Problem. [12] considered the flexible Job shop scheduling problem with transport robots, where each operation can be realized by a subset of machines and adding the transport movement after each machine operation. To solve this problem, an iterative local search algorithm is proposed based on classical exchange, insertion and perturbation moves. Then a simulated annealing schema is used for the acceptance criterion. A hybrid metaheuristic approach is proposed by [37] for the flexible Job Shop problem with transport constraints and bounded processing times. This hybrid approach is composed of a genetic algorithm to solve the assignment problem of operations to machines, and then a tabu search procedure is used to find new improved scheduling solutions. [24] solved the machines and robots simultaneous scheduling problem in flexible manufacturing systems, by adapting a memetic algorithm using a genetic coding containing two parts: a resource selection part for machine operations and a sequencing part for transport operations. [38] considered the job shop scheduling problem with transport robots and bounded processing times. A modified shifting bottleneck procedure is used coupled with a heuristic for assigning and sequencing transportation tasks iteratively.

Furthermore, a newly maturing area of the distributed artificial intelligence is used, providing some effective mechanisms for the management of such dynamic operations in manufacturing environments, such as the multi agent systems. [8] treated the machines and robots scheduling problem in flexible manufacturing systems. They proposed a distributed model based on cooperative agents, composed of five agents: an order-agent, a store-agent, a set of machine-agents and a set of robot-agents, using negotiation between them in order to obtain a best scheduling solution for this problem. [22] formulated the machines and robots scheduling problem in flexible manufacturing systems as a multi-agent system, allowing to realize an Agent-Based Shop Floor Simulator (ABSFSim). This simulator is composed of eight agents classified into three categories: the first category contains agents with a single instance such as the part-generator-agent, the arrival-queue-agent and departure-agent; the second category includes agents with multiple instances and a long lifetime such as the machine-agent, the robot-agent, the node-agent and segment-agent; and the third category contains agents with multiple instances and a short lifetime such as the part-agent. A multi-agent approach is proposed by [13] for robots and machines scheduling problem within a manufacturing system. The proposed multi-agent approach worked under a real-time environment and generated feasible schedules using negotiation/bidding mechanisms between agents. This approach is composed of four agents: a manager-agent, a robot-system-holon, an order-system-holon and a machine-system-holon.

In this paper, we propose a hybridization of two metaheuristics based on clustered holonic multiagent model for the job shop scheduling problem with a single transport robot. This new approach follows two principal hierarchical steps, where a genetic algorithm is applied by a scheduler agent for a global exploration of the search space. Then, a tabu search technique is used by a set of cluster agents to guide the research in promising regions. Numerical tests were made to evaluate the performance of our approach based on the instances of [18], completed by comparisons with other approaches.

The rest of the paper is organized as follows. In Sect. 2, we define the formulation of the FJSPT-SR with its objective function and a simple problem instance. Then, in Sect. 3, we detail the proposed hybrid approach with its holonic multiagent levels. The experimental and comparison results are provided in Sect. 4. Finally, Sect. 5 ends the paper with a conclusion.

2 Problem Formulation

There is a set of n jobs $J = \{J_1, \dots, J_n\}$ to be processed without preemption on a set $M = \{M_0, M_1, \dots, M_m\}$ of $m+1$ machines (M_0 represents the load/unload or LU station from which jobs enter and leave the system). Each job J_i is formed by a sequence of ni operations $\{O_{i,1}, O_{i,2}, \dots, O_{i,ni}\}$ to be performed successively according to the given sequence. For each operation $O_{i,j}$, there is a machine $\mu_{ij} \in \{M_0, \dots, M_m\}$ and a processing time p_{ij} associated with it. In addition, each job J_i $(J_1, \dots, J_n)$ is composed of $ni-1$ transport operations $\{T_{i,1}, T_{i,2}, \dots, T_{i,ni-1}\}$ to be made by a robot R from one machine to another. In fact, for each transport operation $T_{i,j}$ there are two types of movements: travel transport operation and empty transport operation.

Firstly, travel transport operation $t_{\mu_{i,j},\mu_{i,j+1}}$ must be considered for the robot R when an operation $O_{i,j}$ is processed on machine $\mu_{i,j}$ and operation $O_{i,j+1}$ is processed on machine $\mu_{i,j+1}$. These transportation times are job-independent and robot-dependent. Each transportation operation is assumed to be processed by only one transport robot R which can handle at most one job at one time. For convenience, $t_{\mu_{i,j},\mu_{i,j+1}}$ is used to denote both a transportation operation and a transportation time.

Secondly, empty transport operation $t'_{i,j}$ has to be considered while the robot R moves from machine M_i to machine M_j without carrying a job. So, it is possible to assume that $t'_{i,i} = 0$ and $t_{i,j} \geq t'_{i,j}$.

As in job shop problems, we assume that sufficient buffer space exists between machines. This assumption is also stated as an "unlimited input/output buffer capacity". Jobs processed on one machine M_i are assumed to wait until the robot affected to this transport operation is available to do it. No additional time is required to transfer job from machine to the unlimited output buffer. In a similar way, each machine M_i has an unlimited input buffer to store jobs in waiting to be processed by it. All data p_{ij}, $t_{\mu_{i,j},\mu_{i,j+1}}$, $t'_{\mu_{i,j},\mu_{i,j+1}}$ are assumed to be non-negative integers.

The objective is to determine a feasible schedule which minimizes the makespan $Cmax = Max_{j=1,n}\{C_j\}$ where C_j denotes the completion time of the last operation $O_{i,ni}$ of job J_i including the processing times of machine operations and transport operations.

- Cmax makespan

$$Cmax = Max_{j=1,n}\{C_j\} \tag{1}$$

- Mean makespan

$$MeanCmax = D_i = \frac{1}{n}\sum_{i=1}^{n} C_i \tag{2}$$

- Mean tardiness

$$Lateness = L_i = C_i - D_i \tag{3}$$

$$Tardiness = Tard_i = max[L_i, 0] \tag{4}$$

$$MeanTard = \frac{1}{n}\sum_{i=1}^{n} Tard_i \tag{5}$$

3 Hybrid Metaheuristics Based Clustered Holonic Multiagent Model

[16] elaborated a study about the nature of connections between the genetic algorithm and tabu search metaheuristics, searching to show the existing opportunities for creating a hybrid approach with these two standard methods to take advantage of their complementary features and to solve difficult optimization problems. After this pertinent study, the combination of these two metaheuristics has become more well-known in the literature, which has motivated many researchers to try the hybridization of these two methods for the resolution of different complex problems in several areas.

[14] defined a multiagent system as an artificial system composed of a population of autonomous agents, which cooperate with each other to reach common

objectives, while simultaneously each agent pursues individual objectives. Furthermore, a multiagent system is a computational system where two or more agents interact (cooperate or compete, or a combination of them) to achieve some individual or collective goals. The achievement of these goals is beyond the individual capabilities and individual knowledge of each agent [6].

[21] gave the first definition of the term "holon" in the literature, by combining the two Greek words "hol" meaning whole and "on" meaning particle or part. He said that almost everything is both a whole and a part at the same time. In fact, a holon is recursively decomposed at a lower granularity level into a community of other holons to produce a holarchy [9]. Moreover, a holon may be viewed as a sort of recursive agent, which is a super-agent composed of a sub-agents set, where each sub-agent has its own behavior as a complementary part of the whole behaviour of the super-agent. Holons are agents able to show an architectural recursiveness [15].

In this work, we propose a hybrid metaheuristic approach based on clustering processing two general steps: a first step of global exploration using a genetic algorithm to find promising areas in the search space and a clustering operator allowing to regroup them in a set of clusters. In the second step, a tabu search algorithm is applied to find the best individual solution for each cluster. The global process of the proposed approach is implemented in two hierarchical holonic levels adopted by a recursive multiagent model, named a hybrid Genetic Algorithm with Tabu Search based on clustered Holonic Multiagent model (GATS+HM), see Fig. 1. The first holonic level is composed of a Scheduler Agent which is the Master/Super-agent, preparing the best promising regions of the search space, and the second holonic level containing a set of

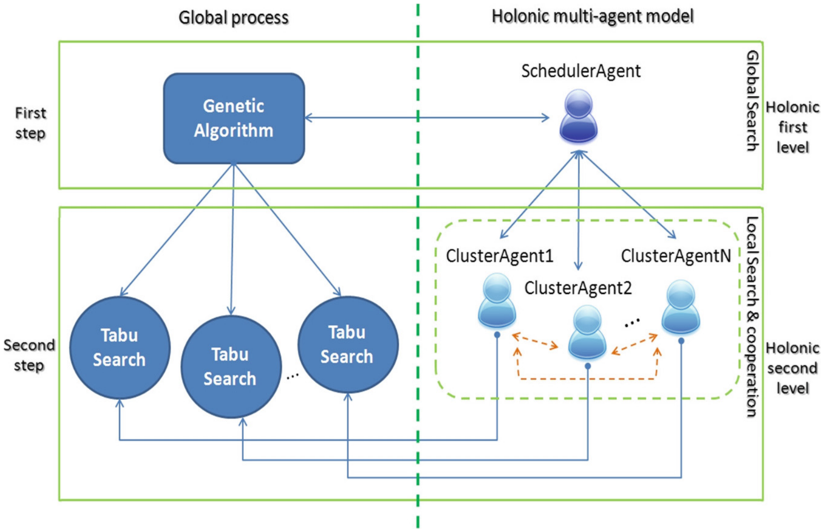


Fig. 1. Hybrid metaheuristics based clustered holonic multiagent model

Cluster Agents which are the Workers/Sub-agents, guiding the search to the global optimum solution of the problem. Each holonic level of this model is responsible for process a step of the hybrid metaheuristic approach and to cooperate between them to attain the global solution of the problem.

The idea of the proposed approach came from that the use of exact methods needed long CPU times to solve big instances of an NP-hard problem which is the simultaneous FJSPT-SR problem is our case. That is why, we thought to use the approximation techniques to generate a set of feasible solutions very close to the optimal solutions of this problem in a reasonable computational time. So, after reviewing the different works made for this problem, we concluded that the standard metaheurstic approaches are well used and presenting many interesting notions such as in the genetic algorithm (GA) and the tabu search (TS), that is why we got the idea to profit from their advantages and trying to combine these two complementary techniques to obtain a new hybrid approach, composed of a set of improved operators, searching to attain more dominant solutions for this problem. Then, after implementing the GA for the global exploration we thought to divide the search space in a set of clusters to select the best obtained solutions preparing the step of hybridization with the TS algorithm. In this position, we said that why not to apply a parallel version of the TS, where here we decided to use the multi-agent systems for this treatment and we added the cooperative behavior between them to not visit the same solution already explored by another agent. In addition, our hybrid metaheursitc approach followed the paradigm of "Master" and "Workers" which are two recursive hierarchical levels adaptable for a holonic multiagent model, where the Scheduler Agent is the Master/Super-agent of its society and the Cluster Agents are its Workers/Sub-agents. At this level, we completed the full architecture of our approach, where firstly, a scheduler agent applied a Neighborhood-based Genetic Algorithm (NGA) for a global exploration of the search space. Then secondly, a set of cluster agents used a tabu search technique to guide the research in promising regions.

3.1 Non Oriented Disjunctive Graph

In this work, we chose to use the disjunctive graph of [18] for the job shop problem with transportation times and one robot. To explain this graph, a sample problem of three jobs and five machines with their transportation times for a single robot R is presented in Table 1.

The disjunctive graph $G = (V_m \cup V_t, C \cup D_m \cup D_r)$, see Fig. 2, is composed of: a set of vertices V_m containing all machine operations, a set of vertices V_t is the set of transport operations obtained by assigning the robot R to each transport operation, and two dummy nodes 0 and $*$. Also, this graph consists of: a set of conjunctions C representing precedence constraints $O_{i,k} \rightarrow t_{\mu_{i,k},\mu_{i,k+1}} \rightarrow O_{i,k+1}$, undirected disjunctions for machines D_m, and undirected disjunctions D_r for the transport robot R. For each job J_i, $ni - 1$ transport operations $t_{\mu_{i,k},\mu_{i,k+1}}$ are introduced including precedence $O_{i,k} \rightarrow t_{\mu_{i,k},\mu_{i,k+1}} \rightarrow O_{i,k+1}$. In fact, the robot R may be considered as an additional "machine" which has to process all these transport operations. The arcs from machine node to transport node are

Table 1. One instance of flexible job shop problem with one robot

			M_1	M_2	M_3	M_4	M_5
Processing times for each job J_i	J_1	$O_{1,1}$	2	9	4	5	1
		$O_{1,2}$	–	6	–	4	–
	J_2	$O_{2,1}$	1	–	5	–	6
		$O_{2,2}$	3	8	6	–	–
		$O_{2,3}$	–	5	9	3	9
	J_3	$O_{3,1}$	–	6	6	–	–
		$O_{3,2}$	3	–	–	5	4

		M1	M2	M3	M4	M5
Transportation times for the robot R	M1	0	1	2	3	4
	M2	1	0	1	2	3
	M3	2	1	0	1	2
	M4	3	2	1	0	1
	M5	4	3	2	1	0

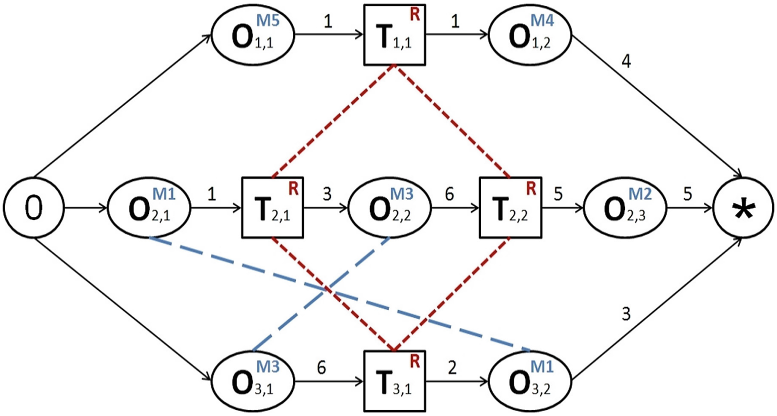


Fig. 2. Non oriented disjunctive graph

weighted with the machine operation durations. Edges between machine operations represent disjunctions for machine operations which have to be processed on the same machine and cannot use it simultaneously.

As for the classical job shop, the conjunctions C model the execution order of operations within each job J_i. In addition to the classical set of undirected machine disjunctions D_m (all pairs of machine operations which have to be processed on the same machine and which are not linked by a directed path), it is necessary to consider the set of undirected robot disjunctions D_r (all pairs of transport operations which have to be performed by the robot R and which are not linked by a directed path). To solve the scheduling problem it is necessary to turn all undirected arcs in $D_m \cup D_r$ into directed ones, and to assign the robot R to each transport operation, where the final graph becomes an oriented disjunctive graph.

3.2 Scheduler Agent

The Scheduler Agent (SA) is responsible for process the first step of the hybrid algorithm by using a genetic algorithm called NGA (Neighborhood-based Genetic Algorithm) to identify areas with high average fitness in the search space during a fixed number of iterations $MaxIter$, see Fig. 3. In fact, the goal of using the NGA is only to explore the search space, but not to find the global solution of the problem. Then, a clustering operator is integrated to divide the best identified areas by the NGA in the search space to different parts where each part is a cluster $CL_i \in CL$ the set of clusters, where $CL = \{CL_1, CL_2, \ldots, CL_N\}$. In addition, this agent plays the role of an interface between the user and the system (initial parameter inputs and final result outputs). According to the number of clusters N obtained after the integration of the clustering operator, the SA creates N Cluster Agents (CAs) preparing the passage to the next step of the global algorithm. After that, the SA remains in a waiting state until the receipt of the best solutions found by the CAs for each cluster CL_i. Finally, it finishes the process by displaying the final solution of the problem.

Individual's Solution Presentation Based Oriented Disjunctive Graph. The Job Shop scheduling Problem with Transportation times and a Single Robot is composed of two sub-problems: firstly the machines and robot selection, secondly the operations scheduling problem, that is why the chromosome representation is encoded in two parts: Machines and Robot Selection part (MRS), and Job and Transport operation Sequence part (JTS), see Fig. 4.

The first part MRS is a vector V_1 with a length L equal to the total number of operations and where each index represents the selected machine or robot to process an operation indicated at position p, see Fig. 4 (a). For example $p = 3$ and

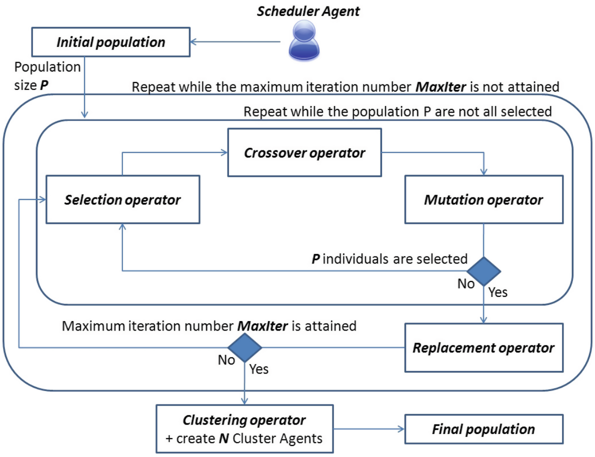
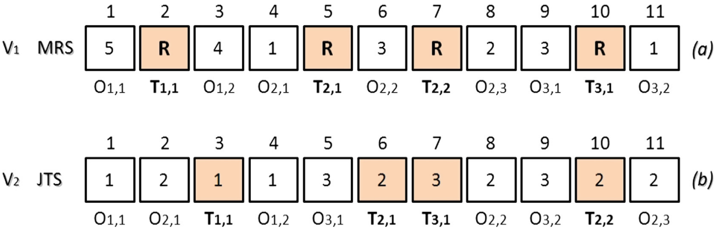

Scheduler Agent
Initial population
Population size P
Repeat while the maximum iteration number MaxIter is not attained
Repeat while the population P are not all selected
Crossover operator
Selection operator
Mutation operator
P individuals are selected
No
Yes
Maximum iteration number MaxIter is attained
Replacement operator
No
Yes
Clustering operator + create N Cluster Agents
Final population

Fig. 3. First step of the global process by the Scheduler Agent



		1	2	3	4	5	6	7	8	9	10	11	
V_1	MRS	5	R	4	1	R	3	R	2	3	R	1	*(a)*
		$O_{1,1}$	**$T_{1,1}$**	$O_{1,2}$	$O_{2,1}$	**$T_{2,1}$**	$O_{2,2}$	**$T_{2,2}$**	$O_{2,3}$	$O_{3,1}$	**$T_{3,1}$**	$O_{3,2}$	

		1	2	3	4	5	6	7	8	9	10	11	
V_2	JTS	1	2	1	1	3	2	3	2	3	2	2	*(b)*
		$O_{1,1}$	$O_{2,1}$	**$T_{1,1}$**	$O_{1,2}$	$O_{3,1}$	**$T_{2,1}$**	**$T_{3,1}$**	$O_{2,2}$	$O_{3,2}$	**$T_{2,2}$**	$O_{2,3}$	

Fig. 4. The chromosome representation of a scheduling solution

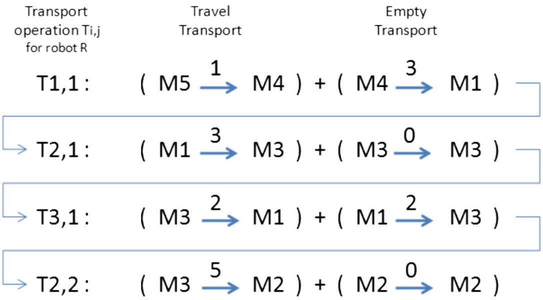

Transport operation Ti,j for robot R
Travel Transport
Empty Transport
T1,1 : (M5 →1 M4) + (M4 →3 M1)
T2,1 : (M1 →3 M3) + (M3 →0 M3)
T3,1 : (M3 →2 M1) + (M1 →2 M3)
T2,2 : (M3 →5 M2) + (M2 →0 M2)

Fig. 5. Final path of the robot R

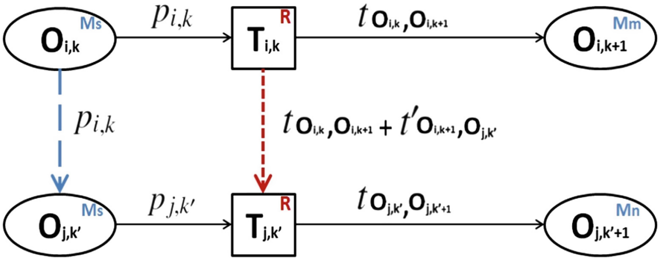


Fig. 6. Example of oriented machine and robot disjunctions

$p = 7$, $V_1(3)$ is the selected machine M_4 for the operation $O_{1,2}$ and $V_1(7)$ is the selected robot R for the operation $T_{2,2}$. The second part JTS is a vector V_2 having the same length of V_1 and where each index represents a machine operation $O_{i,j}$ or a transport operation $T_{i,j}$ according to the predefined operations for each job, see Fig. 4 (b). For example this operation sequence 1-2-1-1-3-2-3-2-3-2-2 can be translated to: $(O_{1,1}, M_5) \rightarrow (O_{2,1}, M_1) \rightarrow (T_{1,1}, R) \rightarrow (O_{1,2}, M_4) \rightarrow (O_{3,1}, M_3) \rightarrow (T_{2,1}, R) \rightarrow (T_{3,1}, R) \rightarrow (O_{2,2}, M_3) \rightarrow (O_{3,2}, M_1) \rightarrow (T_{2,2}, R) \rightarrow (O_{2,3}, M_2)$. In addition, for each job J_i $(J_1, \dots, J_i)$ $ni - 1$ transport operations are generated $T_{1,1}$, $T_{2,1}$, $T_{2,2}$ and $T_{3,1}$, and scheduled following the presented solution in vector JTS, allowing to fix the final path to be considered by the robot R during the shop process, see Fig. 5.

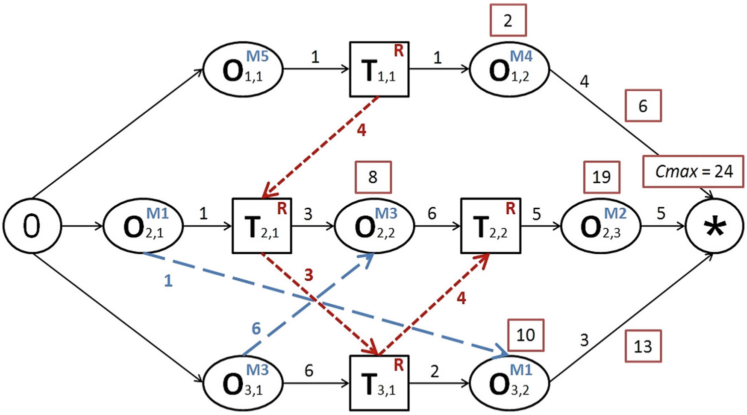


Fig. 7. Oriented disjunctive graph

To model an oriented disjunctive graph we should consider some rules. Let the example in Fig. 6, if the edge is oriented in the direction $O_{i,k} \rightarrow O_{j,k'}$ it gets the weight $p_{i,k}$, else it takes $p_{j,k'}$ in the inverse case. If an arc is added from $T_{i,k}$ to $T_{j,k'}$, it gets the weight $t_{O_{i,k},O_{i,k+1}} + t'_{O_{i,k+1},O_{j,k'}}$ and $t_{O_{j,k'},O_{j,k'+1}} + t'_{O_{j,k'+1},O_{i,k}}$ if it is oriented in the other direction. Thus, basing on [18] we can define a fixed machine selection S_m called directed Machine Disjunctions and a fixed transport selection S_r called directed Transport Disjunctions, with their precedence relations C called operation Conjunctions. So, a fully oriented disjunctive graph can be obtained using $\hat{S} = C \cup S_m \cup S_r$, which is called a complete selection. In fact, the selections of the two sets of disjunctions S_m and S_r with their set of conjunctions C are based on the two proposed vectors MRS and JTS, where MRS allows to present the selected machines to process job operations and the selected robot to process transport operations. JTS presents the execution order of the job and transport operations in their selected machines and robot allowing to fix the final Machine and Transport Disjunctions $S_m \cup S_r$ with their set of Conjunctions C representing the precedence relations between the different operations. The union $C \cup S_m \cup S_r = \hat{S}$ fully describes a solution if the resulting oriented disjunctive graph $G = (V_m, V_t, \hat{S})$ is acyclic. A feasible schedule can be constructed by longest path calculation which permits to obtain the earliest starting time of both machine and transport operations and fully defines a semi-active schedule with the Cmax given by the length of the longest path from node 0 to *, see Fig. 7.

Noting that the chromosome fitness is calculated by $Fitness(i)$ which is the fitness function of each chromosome i and $Cmax(i)$ is its makespan value, where $i \in \{1, \ldots, P\}$ and P is the total population size, see Eq. (6).

$$Fitness(i) = \frac{1}{Cmax(i)} \tag{6}$$

Population Initialization. The initial population is generated randomly following a uniform law and based on a neighborhood parameter to make the individual solutions more diversified and distributed in the search space. In fact, each new solution should have a predefined distance with all the other solutions to be considered as a new member of the initial solution. The used method to determine the neighborhood parameter is inspired from [7], which is based on the permutation level of operations to obtain the distance between two solutions. In fact, the dissimilarity distance is calculated by verifying the difference between two chromosomes in terms of of the placement of each machine operation $O_{i,j}$ on its assigned machine μ_{ij} in the Machines and Robot Selection V_1 (MRS) and the execution order of all the shop operations $O_{i,j}$ and $T_{i,j}$ in the Job and Transport operation Sequence V_2 (JTS). So, if there is a difference in the vector V_1, the distance is incremented by $M(O_{i,j})$ if it is a machine operation ($M(O_{i,j})$ is the number of alternative machines for a machine operation $O_{i,j}$) because it is in the order of O(n). Then, if there is a difference in the vector V_2, the distance is incremented by 1 because it is in the order of O(1). Let $Chrom_1(MRS_1, JTS_1)$ and $Chrom_2(MRS_2, JTS_2)$ two chromosomes of two different scheduling solutions, $M(O_{i,j})$ the number of alternative machines for each machine operation $O_{i,j}$, L is the total number of operations of all jobs and $Dist$ is the dissimilarity distance. The distance is calculated firstly by measuring the difference between the Machines and Robot Selection vectors MRS_1 and MRS_2 which is in order of O(n), then by verifying the execution order difference of the Job and Transport operation Sequence vectors JTS_1 and JTS_2 which is in order of O(1), we give how to proceed in Algorithm 1.

Algorithm 1. How to calculate the dissimilarity distance between two solutions

```
procedure
    $Dist \leftarrow 0, k \leftarrow 1$
    for k from 1 to L do
        if Chrom1($MRS_1$(k)) $\neq$ Chrom2($MRS_2$(k)) and IsMachineOperation($MRS_1$(k)) then
            $Dist \leftarrow Dist + M(O_{i,j})$
        end if
        if Chrom1($JTS_1$(k)) $\neq$ Chrom2($JTS_2$(k)) then
            $Dist \leftarrow Dist + 1$
        end if
    end for
    return $Dist$
end procedure
```

Noting that $Distmax$ is the maximal dissimilarity distance and it is calculated by Eq. (7), representing 100% of difference between two chromosomes.

$$Distmax = \sum_{i=1}^{n} \sum_{i,1}^{i,ni} [M(O_{i,j})] + L \tag{7}$$

Selection Operator. The selection operator is used to select the best parent individuals to prepare them to the crossover step. This operator is based on the fitness function allowing to analyze the quality of each selected solution. But progressively the fitness values will be similar for the most individuals. That is why, we integrate the neighborhood parameter, where we propose a new combined parent selection operator named Fitness-Neighborhood Selection Operator (FNSO) allowing to add the dissimilarity distance parameter to the fitness function to select the best parents for the crossover step. The FNSO chooses in each iteration two parent individuals until engaging all the population to create the next generation. The first parent takes successively in each case a solution i, where $i \in \{1, \ldots, P\}$ and P is the total population size. The second parent obtains its solution j randomly by the roulette wheel selection method based on the two Fitness and Neighborhood parameters relative to the selected first parent, where $j \in \{1, \ldots, P\} \setminus \{i\}$ in the P population and where $j \neq i$. In fact, to use this random method, we should calculate the Fitness-Neighborhood total FN for the population, see Eq. (8), the selection probability sp_k for each individual I_k, see Eq. (9), and the cumulative probability cp_k, see Eq. (10). After that, a random number r will be generated from the uniform range [0,1]. If $r \leq cp_1$ then the second parent takes the first individual I_1, else it gets the kth individual $I_k \in \{I_2, \ldots, I_P\} \setminus \{I_i\}$ and where $cp_{k-1} < r \leq cp_k$. For Eqs. (8), (9) and (10), $k = \{1, 2, \ldots, P\} \setminus \{i\}$.

- The Fitness-Neighborhood total for the population:

$$FN = \sum_{k=1}^{P} [1/(Cmax[k] \times Neighborhood[i][k])] \tag{8}$$

- The selection probability sp_k for each individual I_k:

$$sp_k = \frac{1/(Cmax[k] \times Neighborhood[i][k])}{FN} \tag{9}$$

- The cumulative probability cp_k for each individual I_k:

$$cp_k = \sum_{h=1}^{k} sp_h \tag{10}$$

Crossover Operator. The crossover operator has an important role in the global process, allowing to combine in each case the chromosomes of two parents in order to obtain new individuals and to attain new better parts in the search space. In this work, this operator is applied with two different techniques successively for the parent's chromosome vectors MRS and JTS.

MRS Crossover. A uniform crossover is used to generate in each case a mixed vector between two parent vectors $Parent_1$-MRS_1 and $Parent_2$-MRS_2, allowing to obtain two new children, $Child_1$-MRS'_1 and $Child_2$-MRS'_2. This uniform

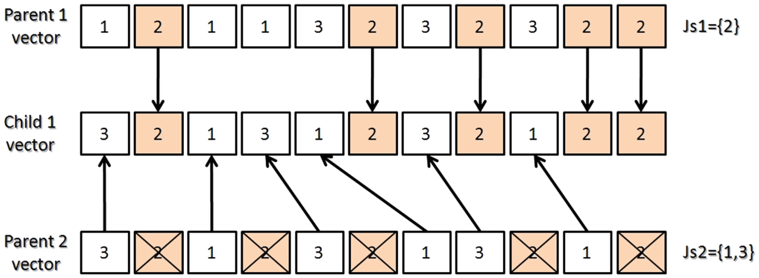


Fig. 8. JTS crossover exemple

crossover is based on two assignment cases, if the generated number r is less than 0.5, the first child $Child_1$ gets the current machine value of $Parent_1$ and the second child $Child_2$ takes the current machine value of $Parent_2$. Else, the two children change their assignment direction, first child $Child_1$ to $Parent_2$ and the second child $Child_2$ to $Parent_1$, we give how to proceed in Algorithm 2.

Algorithm 2. How to generate new children by the MRS crossover

```
procedure
    Generate a random number $r$ in $[0, 1]$
    if $r < 0.5$ then
        $Child_1\text{-}MRS'_1[i] = Parent_1\text{-}MRS_1$
        $Child_2\text{-}MRS'_2[i] = Parent_2\text{-}MRS_2$
    else
        $Child_1\text{-}MRS'_1[i] = Parent_2\text{-}MRS_2$
        $Child_2\text{-}MRS'_2[i] = Parent_1\text{-}MRS_1$
    end if
end procedure
```

JTS Crossover. An improved precedence preserving order-based on crossover (iPOX), inspired from [25], is adapted for the parent operation vector JTS. This iPOX operator is applied via four steps, a first step is selecting two parent operation vectors (JTS_1 and JTS_2) and generating randomly two job subsets Js_1/Js_2 from all jobs. A second step is allowing to copy any element in JTS_1/JTS_2 that belong to Js_1/Js_2 into child individual JTS'_1/JTS'_2 and retain them in the same position. Then the third step deletes the elements that are already in the sub-set Js_1/Js_2 from JTS_1/JTS_2. Finally, fill in order the empty positions in JTS'_1/JTS'_2 with the reminder elements of JTS_2/JTS_1 in the fourth step, see the example in the Fig. 8.

Mutation Operator. The mutation operator is integrated to promote the children generation diversity. In fact, this operator is applied on the chromosomes

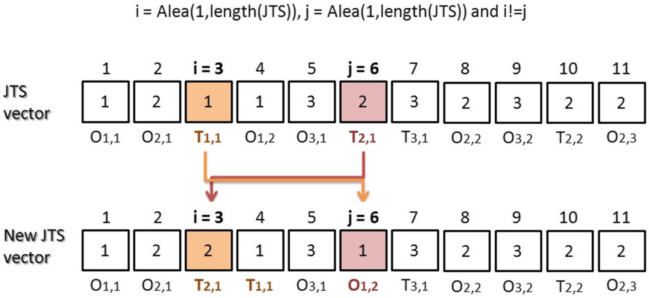

i = Alea(1,length(JTS)), j = Alea(1,length(JTS)) and i!=j
JTS vector
1 2 i = 3 4 5 j = 6 7 8 9 10 11
1 2 1 1 3 2 3 2 3 2 2
O1,1 O2,1 T1,1 O1,2 O3,1 T2,1 T3,1 O2,2 O3,2 T2,2 O2,3
New JTS vector
1 2 i = 3 4 5 j = 6 7 8 9 10 11
1 2 2 1 3 1 3 2 3 2 2
O1,1 O2,1 T2,1 T1,1 O3,1 O1,2 T3,1 O2,2 O3,2 T2,2 O2,3

Fig. 9. JTS mutation exemple

of the newly-generated children by the crossover operator. Also, each part of a child chromosome MRS and JTS has separately its own mutation technique.

MRS Mutation. This first operator uses a random selection of a transport operation index from the vector MRS. Then, it replaces the current number in the selected index by another belonging to the alternative set of machines.

JTS Mutation. This operator selects randomly two indexes $index_1$ and $index_2$ from the vector JTS. Next, it changes the position of the job number in the $index_1$ to the second $index_2$ and inversely, see Fig. 9.

Replacement Operator. The replacement operator has an important role to prepare the remaining surviving population to be considered for the next iterations. This operator replaces in each case a parent by one of its children which has the best fitness in its current family.

Clustering Operator. By finishing the maximum iteration number $MaxIter$ of the genetic algorithm, the Scheduler Agent applies a clustering operator using the hierarchical clustering algorithm of [19] to divide the final population into N Clusters, see Fig. 10, to be treated by the Cluster Agents in the second step of the global process. The clustering operator is based on the neighbourhood parameter which is the dissimilarity distance between individuals. The clustering operator starts by assigning each individual $Indiv(i)$ to a cluster CL_i, so if we have P individuals, we have now P clusters containing just one individual in each of them. For each case, we fix an individual $Indiv(i)$ and we verify successively for each next individual $Indiv(j)$ from the remaining population (where i and $j \in \{1, \ldots, P\}, i \neq j$) if the dissimilarity distance $Dist$ between $Indiv(i)$ and $Indiv(j)$ is less than or equal to a fixed threshold $Distfix$ (representing a percentage of difference X% relatively to $Distmax$, see Eq. (11)) and where $Cluster(Indiv(i)) \neq Cluster(Indiv(j))$. If it is the case, $Merge(Cluster(Indiv(i)), Cluster(Indiv(j)))$, else continue the search for new

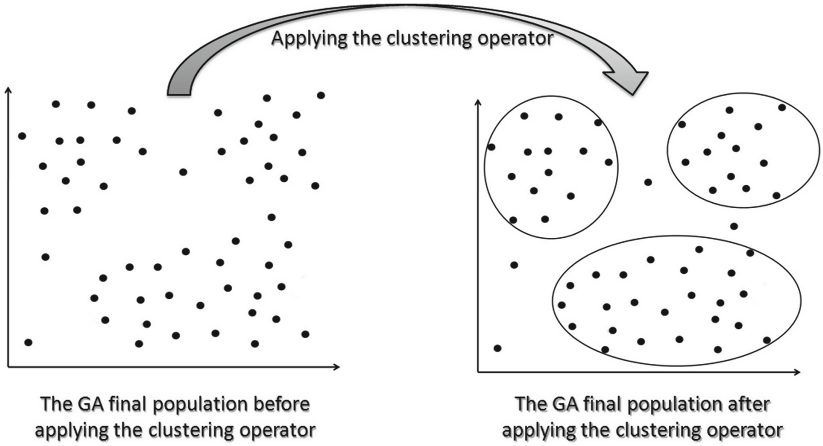

Applying the clustering operator
The GA final population before applying the clustering operator
The GA final population after applying the clustering operator

Fig. 10. The final population transformation by applying the clustering operator

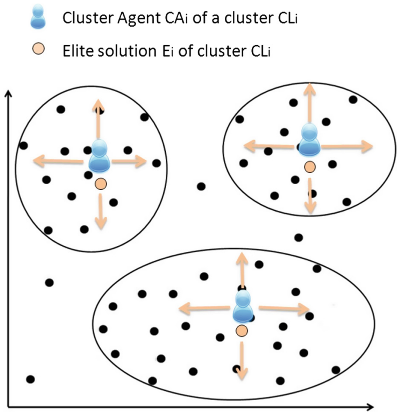

Cluster Agent CAi of a cluster CLi
Elite solution Ei of cluster CLi

Fig. 11. Distribution of the Cluster Agents in the different clusters of the search space

combination with the remaining individuals. The stopping condition is by browsing all the population individuals, where we obtained at the end N Clusters.

$$Distfix = Distmax \times X\% \tag{11}$$

3.3 Cluster Agents

Each Cluster Agent CA_i is responsible for apply successively to each cluster CL_i a local search technique which is the Tabu Search algorithm to guide the research in promising regions of the search space and to improve the quality of the final population of the genetic algorithm. In fact, this local search is executed simultaneously by the set of the CAs agents, where each CA starts the research from an elite solution of its cluster searching to attain new more dominant individual solutions separately in its assigned cluster CL_i, see Fig. 11. The used Tabu Search algorithm is based on an intensification technique allowing to start the research from an elite solution in a cluster CL_i (a promising part in the search space) in order to collect new scheduling sequence minimizing the makespan. Let E the elite solution of a cluster CL_i, $E' \in N(E)$ is a neighbor of the elite solution E, GL_i is the Global List of each CA_i to receive new found elite solutions by the remaining CAs, each CL_i plays the role of the tabu list with a dynamic length and Cmax is the makespan of the obtained solution. So, the search process of this local search starts from an elite solution E using the move and insert method of [28], where each Cluster Agent CA_i changes the position of a machine operation $O_{i,j}$ from a machine M_m to another machine M_n belonging to the alternative set of machines, and modifies the execution order of an operation ($O_{i,j}$ if machine operation or $T_{i,j}$ if transport operation) from an index i to another index k in the vector JTS, searching to generate new scheduling combination $E' \in N(E)$. After that, verifying if the makespan value of this new generated solution $Cmax(E')$ dominates $Cmax(E)$ ($Cmax(E') < Cmax(E)$), and if it is the case CA_i saves E' in its tabu list (which is CL_i) and sends it to all the other CAs agents to be placed in their Global Lists $GLs(E', CA_i)$, to ensure that it will not be used again by them as a search point. Else continues the neighborhood search from the current solution E. The stopping condition is by attaining the maximum allowed number of neighbors for a solution E without improvement. We give how to proceed in Algorithm 3.

Algorithm 3. The local search process

```
procedure
    $E \leftarrow$ Elite($CL_i$)
    while N($E$) $\neq \emptyset$ do
        $E' \leftarrow$ Move-and-insert($E$) | $E' \in$ N($E$) | $E' \notin CL_i$
        if Cmax($E'$) < Cmax($E$) and $E' \notin GL_i$ then
            $E \leftarrow E'$
            $CL_i \leftarrow E'$
            Send-to-all($E'$, $CA_i$)
        end if
    end while
    return $E$
end procedure
```

By finishing this local search step, the CAs agents terminate the process by sending their last best solutions to the SA agent, which considers the best one of them as the global solution for the FJSPT-SR, see Fig. 12.

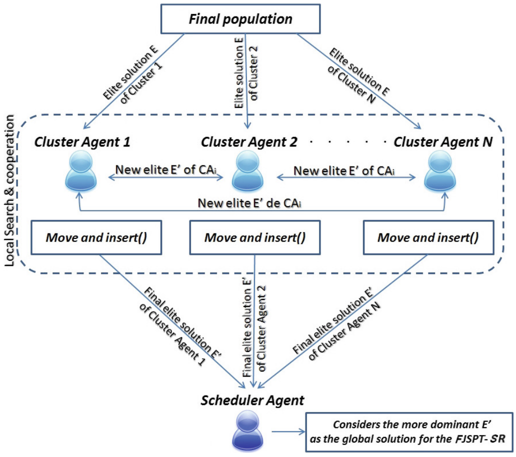

Final population
Elite solution E of Cluster 1
Elite solution E of Cluster 2
Elite solution E of Cluster N
Local Search & cooperation
Cluster Agent 1
Cluster Agent 2
Cluster Agent N
New elite E' of CAi
New elite E' of CAi
New elite E' de CAi
Move and insert()
Move and insert()
Move and insert()
Final elite solution E' of Cluster Agent 1
Final elite solution E' of Cluster Agent 2
Final elite solution E' of Cluster Agent N
Scheduler Agent
Considers the more dominant E' as the global solution for the FJSPT- SR

Fig. 12. Second step of the global process by the Cluster Agents

4 Experimental Results

4.1 Experimental Setup

The proposed GATS+HM is implemented in java language on a 2.10 GHz Intel Core 2 Duo processor and 3 Gb of RAM memory, where we use the Integrated Development Environment (IDE) *Eclipse* to code the algorithm and the multiagent platform *Jade* [4] to create the different agents of our holonic model. To evaluate its efficiency, numerical tests are made based on the benchmark instances of [18] from the literature of the FJSPT-SR, which consists of two sets P1(6×6) and P2(10×10) inspired from [29]. This shop problem considers a single moving robot for all transport operations, where various test instances were obtained by adding transportation and empty moving times with different characteristics.

Due to the non-deterministic nature of the proposed approach, we run it ten independent times for each case of the [18] data instances in order to obtain significant results. The used parameter settings for our algorithm are adjusted experimentally and presented as follow: the crossover probability = 1.0, the mutation probability = 0.5, the maximum number of iterations = 1000 and the population size = 200. The computational results are presented by six metrics as follows:

- The best makespan, the CPU time of our GATS+HM in minutes and the gap between our approach and the best results of the literature in Table 2. The gap values are calculated by Eq. (12), where the $Mk_{\mathbf{O}}$ is the makespan obtained by **O**ur approach and $Mk_{\mathbf{C}}$ is the makespan of one of the chosen algorithms for **C**omparisons.

$$Gap = [(Mk_{\mathbf{O}} - Mk_{\mathbf{C}})/Mk_{\mathbf{C}}] \times 100\% \tag{12}$$

- The mean makespan and mean tardiness, see Eqs. (2) and (5), for the selected set of [18] data instances in Table 3.

4.2 Experimental Comparisons

To show the efficiency of our GATS+HM approach, we compare its obtained results from the [18] benchmark instances with other algorithms from the literature of the FJSPT-SR, which have obtained the best upper bounds for this problem. The chosen algorithms are: the lower bound (LB_H) and the one-stage approach (UB_{one}) of [18] which presented the first results for their proposed instances; the genetic algorithm-tabu search procedure (GATS) of [37], the hybrid memetic algorithm (BFS) of [24] and the shifting bottleneck (SBN) of [38] which are three recent metaheuristic approaches.

From Table 2, the comparison results show that the GATS+HM obtains twelve out of fifteen best results for the [18] instances, where we attain ten new upper bounds and two similar optimal solutions. Indeed, our algorithm outperforms the UBone in eleven out of fifteen instances with a maximum gap of −7,80% for the P02-T5-t2 instance, and it gets slightly worse results for three instances with a maximum gap of 1,21% for the P02-D3-d1 instance. Moreover, our GATS+HM outperforms the BFS in twelve out of fifteen instances with a maximum gap of −4,13% for the P02-T2-t1 instance, and it gets one bad result for the P01-D2-d1 instance with a gap of 0,68%. For the comparison with the GATS, the GATS+HM obtains fourteen out of fifteen best results with a maximum gap of −30,52% for the P02-D2-d1 instance. In addition, we dominate the SBN in all the fifteen instances with a gap varying from −3,16% for the P01-T3-t0 instance to −46,15% for the P01-D1-d1 instance.

Table 2. Makespan results for [18] data instances

Instance	LB_H	Gap	UB_{one}	Gap	BFS	Gap	GATS	Gap	SBN	Gap	GATS+HM	CPU Time
P01.D1-d1	**82**	2,44	87	-3,45	87	-3,45	96	-12,50	156	-46,15	**84**	0,30
P01.D1-t1	**77**	2,60	81	-2,47	81	-2,47	83	-4,82	93	-15,05	**79**	0,27
P01.D2-d1	**147**	1,36	**148**	0,68	**148**	0,68	155	-3,87	158	-5,70	149	0,29
P01.D3-d1	**213**	0,00	217	-1,84	**213**	0,00	220	-3,18	224	-4,91	**213**	0,28
P01.T2-t1	**71**	1,41	74	-2,70	74	-2,70	79	-8,86	93	-22,58	**72**	0,26
P01.T3-t0	**92**	0,00	**92**	0,00	**92**	0,00	**92**	0,00	95	-3,16	**92**	0,19
P02.D1-d1	**880**	10,57	1044	-6,80	1012	-3,85	1339	-27,33	1454	-33,08	**973**	6,15
P02.D1-t0	**880**	12,50	1042	-4,99	1017	-2,65	1352	-26,78	1430	-30,77	**990**	4,26
P02.D1-t1	**880**	11,36	1016	-3,54	983	-0,31	1337	-26,70	1490	-34,23	**980**	4,56
P02.D2-d1	**892**	12,56	1070	-6,17	1045	-3,92	1445	-30,52	1576	-36,29	**1004**	8,53
P02.D3-d1	**906**	19,54	**1070**	1,21	1100	-1,55	1516	-28,56	1567	-30,89	1083	11,45
P02.D5-t2	**1167**	13,71	**1325**	0,15	1361	-2,50	1689	-21,43	1576	-15,80	1327	13,15
P02.T1-t1	**874**	8,35	1006	-5,86	978	-3,17	1322	-28,37	1376	-31,18	**947**	7,14
P02.T2-t1	**880**	8,18	1015	-6,21	993	-4,13	1279	-25,57	1394	-31,71	**952**	7,31
P02.T5-t2	**898**	13,14	1102	-7,80	1022	-0,59	1339	-24,12	1413	-28,10	**1016**	9,25

Table 3. Mean makespan and mean tardiness results for [18] data instances

LB_H		UBone		BFS		GATS		SBN		GATS+HM	
Mean Cmax	Mean Tard	Mean Cmax	Mean Tard	Mean Cmax	Mean Tard	Mean Cmax	Mean Tard	Mean Cmax	Mean Tard	Mean Cmax	Mean Tard
595.94	**192.91**	692.60	230.44	680.40	225.83	889.54	307.47	939.67	321.26	**664.07**	**219.69**

From Table 3, it can be seen that the mean results obtained by our GATS+HM for the [18] instances are the closest ones to the LB mean results in terms of the mean makespan and the mean tardiness, and also are the best obtained ones comparing to the other upper bound approaches UBone, BFS, GATS and SBN. Consequently, we can distinguish the efficiency of our approach not only on each single instance as in Table 2, but also on the mean result of the treated set of instances.

By evaluating the computational time in few minutes and the comparison results of our approach in terms of the makespan, the mean makespan and the mean tardiness, we can distinguish the efficiency of the new proposed GATS+HM relatively to the literature of the FJSPT-SR. This efficiency is explained by the flexible selection of the promising parts of the search space by the clustering operator after the genetic algorithm process and by applying the intensification technique of the tabu search allowing to start from a set of elite solutions to attain new more dominant solutions.

5 Conclusion

In this paper, we present a new metaheuristic hybridization approach based on clustered holonic multiagent model, called GATS+HM, for the flexible job shop scheduling problem with transportation times and a single robot (FJSPT-SR). In this approach, a neighborhood-based genetic algorithm is adapted by a scheduler agent for a global exploration of the search space. Then, a local search technique is applied by a set of cluster agents to guide the research in promising regions of the search space and to improve the quality of the final population. To measure its performance, numerical tests are made using benchmark data instances from the literature of FJSPT-SR, and where new upper bounds are found showing the effectiveness of the presented approach. In the future work, we will search to treat other extensions of the FJSPT-SR, such as the general case of this problem where a set of robots can be used for the transport oprations, and by considering the machine assignment problem for each operation in the shop process. So, the problem becomes a Flexible Job Shop scheduling Problem with Transportation times and Many Robots (FJSPT-MR). Thus, we will make improvements in the chromosome first part MRS to adapt it to this new transformation and study its effects on the makespan.

References

1. Abdelmaguid, T.F., Nassef, A.O., Kamal, B.A., Hassan, M.F.: A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. Int. J. Prod. Res. **42**(2), 267–281 (2004)
2. Anwar, M.F., Nagi, R.: Integrated scheduling of material handling and manufacturing activities for just-in-time production of complex assemblies. Int. J. Prod. Res. **36**(3), 653–681 (1998)
3. Babu, A.G., Jerald, J., Haq, A.N., Luxmi, V.M., Vigneswaralu, T.P.: Scheduling of machines and automated guided vehicles in fms using differential evolution. Int. J. Prod. Res. **48**(16), 4683–4699 (2010)
4. Bellifemine, F., Poggi, A., Rimassa, G.: JADE - A FIPA-compliant agent framework. In: Proceedings of the Fourth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agent Technology, pp. 97–108, April 1999
5. Bilge, U., Ulusoy, G.: A time window approach to simultaneous scheduling of machines and material handling system in an FMS. Oper. Res. **43**(6), 1058–1070 (1995)
6. Botti, V., Giret, A.: ANEMONA: A Multi-agent Methodology for Holonic Manufacturing Systems. Springer, London (2008). https://doi.org/10.1007/978-1-84800-310-1
7. Bozejko, W., Uchronski, M., Wodecki, M.: The new golf neighborhood for the flexible job shop problem. In: Proceedings of the International Conference on Computational Science, pp. 289–296, May 2010
8. Braga, R.A.M., Rossetti, R.J.F., Reis, L.P., Oliveira, E.C.: Applying multi-agent systems to simulate dynamic control in flexible manufacturing scenarios. In: European Meeting on Cybernetics and Systems Research, vol. 2, pp. 488–493. Austrian Society for Cybernetic Studies (2008)
9. Calabrese, M.: Hierarchical-granularity holonic modelling. Doctoral thesis, Universita degli Studi di Milano, Milano, Italy, March 2011
10. Caumond, A., Lacomme, P., Moukrim, A., Tchernev, N.: An MILP for scheduling problems in an FMS with one vehicle. Eur. J. Oper. Res. **199**(3), 706–722 (2009)
11. Deroussi, L., Gourgand, M., Tchernev, N.: A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles. Int. J. Prod. Res. **46**(8), 2143–2164 (2008)
12. Deroussi, L., Norre, S.: Simultaneous scheduling of machines and vehicles for the flexible job shop problem. In: International Conference on Metaheuristics and Nature Inspired Computing, pp. 1–2 (2010)
13. Erol, R., Sahin, C., Baykasoglu, A., Kaplanoglu, V.: A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems. Appl. Soft Comput. **12**(6), 1720–1732 (2012)
14. Ferber, J.: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence, 1st edn. Addison-Wesley Longman Publishing Co. Inc., Boston (1999)
15. Giret, A., Botti, V.: Holons and agents. J. Intell. Manuf. **15**(5), 645–659 (2004)
16. Glover, F., Kelly, J.P., Laguna, M.: Genetic algorithms and tabu search: hybrids for optimization. Comput. Oper. Res. **22**(1), 111–134 (1995)
17. Hurink, J., Knust, S.: A tabu search algorithm for scheduling a single robot in a job-shop environment. Discrete Appl. Math. **119**(1–2), 181–203 (2002)
18. Hurink, J., Knust, S.: Tabu search algorithms for job-shop problems with a single transport robot. Eur. J. Oper. Res. **162**(1), 99–111 (2005)

19. Johnson, S.C.: Hierarchical clustering schemes. Psychometrika **32**(3), 241–254 (1967)
20. Jones, A., Rabelo, L.C.: Survey of job shop scheduling techniques. Technical report, National Institute of Standards and Technology, Gaithersburg, USA (1998)
21. Koestler, A.: The Ghost in the Machine, 1st edn. Hutchinson, London (1967)
22. Komma, V.R., Jain, P.K., Mehta, N.K.: An approach for agent modeling in manufacturing on JADE reactive architecture. Int. J. Adv. Manuf. Technol. **52**(9–12), 1079–1090 (2011)
23. Lacomme, P., Larabi, M., Tchernev, N.: A disjunctive graph for the job-shop with several robots. In: Multidisciplinary International Conference on Scheduling: Theory and Applications, pp. 285–292. MISTA, Paris, France (2007)
24. Lacomme, P., Larabi, M., Tchernev, N.: Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. Int. J. Prod. Econ. **143**(1), 24–34 (2013)
25. Lee, K., Yamakawa, T., Lee, K.M.: A genetic algorithm for general machine scheduling problems. In: Proceedings of the second IEEE international Conference on Knowledge-Based Intelligent Electronic Systems, pp. 60–66, April 1998
26. Lenstra, J.K., Kan, A.H.G.R.: Computational complexity of scheduling under precedence constraints. Ann. Discrete Math. **4**, 121–140 (1979)
27. Lenstra, J.K., Kan, A.H.G.R.: Complexity of vehicle routing and scheduling problems. Networks **11**(2), 221–227 (1981)
28. Mastrolilli, M., Gambardella, L.: Effective neighbourhood functions for the flexible job shop problem. J. Sched. **3**(1), 3–20 (2000)
29. Muth, J.F., Thompson, G.L.: Industrial Scheduling. International series in management. Prentice-Hall, Englewood Cliffs (1963)
30. Nouri, H.E., Driss, O.B., Ghédira, K.: Solving the flexible job shop problem by hybrid metaheuristics-based multiagent model. J. Ind. Eng. Int. **14**(1), 1–14 (2017)
31. Pundit, R., Palekar, U.: Job shop scheduling with explicit material handling considerations. Technical report, Univ. of Illinois at Urbana-Champaign, Dept. of M. and I.E (1990)
32. Raman, N., Talbot, F.B., Rachamadgu, R.V.: Simultaneous scheduling of machines and material handling devices in automated manufacturing. In: Proceedings of the 2nd ORSA/TIMS Conference on Flexible Manufacturing Systems, pp. 455–466 (1986)
33. Reddy, B.S.P., Rao, C.S.P.: A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS. Int. J. Adv. Manuf. Technol. **31**(5–6), 602–613 (2006)
34. Sonmez, A.I., Baykasoglu, A.: A new dynamic programming formulation of (n×m) flow shop sequencing problems with due dates. Int. J. Prod. Res. **36**(8), 2269–2283 (1998)
35. Storn, R., Price, K.: Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute, Berkeley (1995)
36. Ulusoy, G., Erifolu, F.S., Bilge, U.: A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles. Comput. Oper. Res. **24**(3), 335–351 (1997)

37. Zhang, Q., Manier, H., Manier, M.A.: A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. Comput. Oper. Res. **39**(7), 1713–1723 (2012)
38. Zhang, Q., Manier, H., Manier, M.A.: A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints. Int. J. Prod. Res. **52**(4), 985–1002 (2014)

Can Evolution Strategies Benefit from Shrinkage Estimators?

Silja Meyer-Nieberg(✉) and Erik Kropat

Department of Computer Science, Universität der Bundeswehr München,
Werner-Heisenberg Weg 37, 85577 Neubiberg, Germany
{silja.meyer-nieberg,erik.kropat}@unibw.de

Abstract. Evolution strategies are evolutionary algorithms usually applied for solving continuous optimization tasks. As they rely on mutation as one of the main search operators, the control and the adaptation of this process is of high importance. This paper discusses the covariance matrix adaptation in evolution strategies, a central and essential mechanism for the search. The current form bases the estimation of the covariance matrix on small samples sizes compared to the search space dimension which is known to be problematic. This leads to the question, whether the performance of the evolutionary algorithms could be improved if other estimators were utilized. In statistics, several alternative approaches have been considered. Up to now, they have only been seldom applied in evolutionary computation. The paper investigates whether evolution strategies may benefit from linear shrinkage estimators. Several shrinkage targets are considered, integrated in the so-called CMSA-ES, and analyzed experimentally with a special focus on the shrinkage intensity.

1 Introduction

Evolutionary algorithms (EAs) are metaheuristics which are based on the principles of natural evolution. As such, they use recombination and mutation to create new search points and perform selection in order to determine which points may pass on their characteristics to succeeding generations. Research in EAs has a long tradition: The first evolutionary algorithms have been introduced in the 1960 s. Today, they represent one of the major classes in natural computation. Five main groups exist: genetic algorithms, genetic programming, evolutionary programming, differential evolution, and evolution strategies. The present article focuses on the last class, *evolution strategies* (ESs), which date back to the 70 s and are – at least today – mainly applied for continuous optimization. In this area, they have been established as well-performing black-box optimization methods, see e.g. [15]. Their main search operator is mutation in contrast to genetic algorithms which favor recombination.

Evolution strategies, see for example [1,3,32] for an introduction, operate with a multivariate normal distribution to generate new search points. The main parameters of the search distribution, the mean $\mathbf{m}$ and the covariance matrix

© Springer International Publishing AG, part of Springer Nature 2018
N. T. Nguyen et al. (Eds.): TCCI XXVIII, LNCS 10780, pp. 116–142, 2018.
https://doi.org/10.1007/978-3-319-78301-7_6

$\sigma^2\mathbf{C}$, must be adapted during the run of an evolution strategy. If they remain constant, the ES may exhibit either only a slow convergence behavior or it may fail in its optimization task entirely. In order to ensure that the mutation control parameters are suited to the current fitness landscape of the function to be optimized, the scale as well as the directions of the mutations must be adapted. For this reason, research in evolution strategies often centers on control techniques for the mutation process with a strong focus on the covariance matrix. Nearly all approaches estimate the current covariance based on the sample. Relying on the sample covariance matrix leads to an ill-posed estimation task in the case of evolution strategies, however: Due to efficiency, the sample size is typically small w.r.t. the search space dimension. This results in a well-known problem in statistics: The estimate may differ considerably from the underlying true covariance [35,36]. This may be the reason why nearly all current techniques introduce additional correction or regularization terms for example by falling back to the previous covariance matrix and/or by strengthening certain promising directions. Interestingly, these procedures are reminiscent of *shrinkage estimation* in statistics which represents one technique to cope with a poor sample covariance estimate. These similarities lead to the research question of the present paper: If evolution strategies perform a kind of implicit shrinkage, can they profit from the introduction of explicit shrinkage operators?

So far, shrinkage and other covariance matrix estimators have been applied remarkably seldom in the area of evolutionary computation. A literature review resulted in only two papers aside from our previous approaches: The first by Dong and Yao explored an application in the case of Gaussian estimation of distribution algorithms [8]. They faced the problem that the learning of the covariance matrix during the run lead to non positive definite matrices. For this reason, a shrinkage procedure was applied to “repair” the covariance matrix. The approach was similar to in [20] with the exception of an adaptable shrinkage intensity. More recently, Kramer considered a Ledoit-Wolf-estimator based on [19] for an evolution strategy which does not follow a population-based approach but uses a variant of the single-point elistist $(1+1)-$ES. For this reason, the covariance matrix adaptation has to consider past search points and corrects the estimate with shrinkage [17].

The current analysis extends the work carried out in [27,28] and augments the investigation conducted in [25,29] for the case of thresholding estimators. [27,28] presented the first approaches to apply Ledoit-Wolf shrinkage estimators in evolution strategies. In a proof of concept, the shrinkage estimators were combined with an approach stemming from a maximum entropy covariance selection principle. Here, the work is extended by considering several mixture matrices, targets, and choices for the shrinkage intensity which are compared to the original version of the underlying ES variant, the CMSA-ES [5], for noise-free and for noisy optimization.

The paper is structured as follows: First, a brief introduction into evolution strategies with covariance matrix adaptation is provided. Afterwards, we focus on the problem of estimating high-dimensional covariance matrices.

Several shrinkage targets are introduced and their integration into evolution strategies is described. The strategies are assessed and compared to the original ES version in the experimental sections. The paper ends with the conclusions and an outlook regarding open research points.

2 Evolution Strategies: A Short Introduction

Let $f : \mathbb{R}^N \to \mathbb{R}$ be a continuous function that allows only the evaluation of the function itself but not the derivation of higher order information. This is the area of black-box optimization, where metaheuristics as evolution strategies and similar approaches are often applied. *Evolution strategies* (ESs) are stochastic optimization methods that usually operate with a sample or population of search points also called *candidate solutions*. They distinguish between a population of μ parents and λ offspring. In many applications in continuous search spaces, the parent population is discarded after the offspring have been created. Therefore, $\lambda > \mu$ is required. Evolution strategies use a multivariate normal distribution with mean $\mathbf{m}^{(g)}$ and covariance matrix $\left(\sigma^{(g)}\right)^2 \mathbf{C}^{(g)}$ to create new search points. The mean represents the recombination process and is obtained as the (weighted) centroid of the parent population. The covariance matrix is updated by following one of the established techniques [5,12]. Sampling λ times from the normal distribution results in the offspring population

$$\mathbf{x}_l = \mathbf{m}^{(g)} + \sigma^{(g)}\mathcal{N}(0, \mathbf{C}^{(g)}),\ l = 1, \ldots, \lambda. \tag{1}$$

Afterwards, the new search points are evaluated using the function f to be optimized. Evolution strategies then apply deterministic selection in order to determine the next parent population and chose the μ best of the λ offspring.

As stated earlier, the parameters of the normal distribution must be adapted in order to allow progress towards the optimal point. Here, the covariance matrix is of great importance and adaptation techniques have received a lot of attention in work on evolution strategies (see [12,24]). The investigation in this paper centers on the *covariance matrix self-adaptation evolution strategy* (CMSA-ES) [5]. The CMSA-ES divides the adaptation of the covariance $(\sigma^{(g)})^2\mathbf{C}^{(g)}$ into two main procedures: the covariance matrix update for $\mathbf{C}^{(g)}$ and the adaptation of the scale factor $\sigma^{(g)}$. The scale factor is also often called step-size or *mutation strength*. Following established practice in evolution strategies, the matrix $\mathbf{C}^{(g)}$ will be referred to as the *covariance matrix* in the remainder of the paper.

2.1 Adaptation I: Covariance Matrix Update

As it is the case for most techniques, the covariance matrix update of the CMSA-ES is based on the sample estimate, that is, on the μ best offspring. Considering only the superior candidate solutions shall introduce a bias towards good search regions. Let $\mathbf{x}_{m:\lambda}$ denote the mth best of the λ offspring w.r.t. the fitness and let

$$\mathbf{z}_{m:\lambda}^{(g+1)} := \frac{1}{\sigma^{(g)}}\left(\mathbf{x}_{m:\lambda}^{(g+1)} - \mathbf{m}^{(g)}\right), \tag{2}$$

stand for its normalization. The estimate then reads

$$\mathbf{C}_{\mu}^{(g+1)} := \sum_{m=1}^{\mu} w_m \mathbf{z}_{m:\lambda}^{(g+1)} (\mathbf{z}_{m:\lambda}^{(g+1)})^{\mathrm{T}} \tag{3}$$

with the weights usually set to $w_m = 1/\mu$ for the CMSA-ES [5]. Please note that the mean $\mathbf{m}^{(g)}$ is known, thus, the number of degrees of freedom remains equal to μ. The sample covariance is then combined with the old covariance, resulting in

$$\mathbf{C}^{(g+1)} := (1 - \frac{1}{c_\tau})\mathbf{C}^{(g)} + \frac{1}{c_\tau}\mathbf{C}_{\mu}^{(g+1)}. \tag{4}$$

The parameter $c_\tau \in (0, 1)$,

$$c_\tau = 1 + \frac{N(N+1)}{2\mu}, \tag{5}$$

see [5] increases with the search space dimension and decreases with the population size.

2.2 Adaptation II: Self-adaptation

In addition to the covariance matrix update, the CMSA-ES applies *self-adaptation* to control the mutation strength $\sigma^{(g)}$. Self-Adaptation has been developed by Rechenberg [31] and Schwefel [33]. It takes place at the level of the individuals meaning that each population member operates with a distinct set. The strategy parameters are adapted by using similar evolutionary principles as for the main evolutionary algorithm. In other words, they also undergo recombination and mutation processes. Afterwards, they are used in the mutation of the search space position. The influence on the selection is indirect: Self-adaptation is based on the assumption of a stochastic linkage between good objective values and appropriately tuned strategy parameters.

In the case of the CMSA-ES, the mutation process of the mutation strength is realized with the help of the log-normal distribution following

$$\sigma_l^{(g)} = \sigma^{(g)} \exp(\tau \mathcal{N}(0, 1)). \tag{6}$$

The parameter τ, the *learning rate*, should scale with $1/\sqrt{2N}$ [23]. Self-adaptation with recombination has been shown as robust against noise [2] and is therefore considered in this paper. In this case, the variable $\sigma^{(g)}$ in (6) is the result of the recombination of the mutation strengths. Here, the same recombination type as previously may be used, that is, $\sigma^{(g+1)} = \sum w_m \sigma_{m:\lambda}$ with $\sigma_{m:\lambda}$ standing for the mutation strength associated with the mth best individual. Figure 1 summarizes the main steps of the covariance matrix self-adaptation evolution strategy.

```
Require: $\lambda$, $\mu$, $\mathbf{C}^{(0)}$, $\mathbf{m}^{(0)}$, $\sigma^{(0)}$, $\tau$, $c_\tau$
g = 0
while termination criteria not met do
    for $l = 1$ to $\lambda$ do
        $\sigma_l = \sigma^{(g)} \exp(\tau \mathcal{N}(0,1))$
        $\mathbf{x}_l = \mathbf{m}^{(g)} + \sigma_l \boldsymbol{\mathcal{N}}(0, \mathbf{C}^{(g)})$
        $f_l = f(\mathbf{x}_l)$
    end for
    Select $(\mathbf{x}_{1:\lambda}, \sigma_{1:\lambda}), \dots, (\mathbf{x}_{\mu:\lambda}, \sigma_{\mu:\lambda})$ according to their fitness $f_l$
    $\mathbf{m}^{(g+1)} = \sum_{m=1}^{\mu} w_m \mathbf{x}_{m:\lambda}$
    $\sigma^{(g+1)} = \sum_{m=1}^{\mu} w_m \sigma_{m:\lambda}$
    $\mathbf{z}_{m;\lambda} = \frac{\mathbf{x}_{m;\lambda} - \mathbf{m}^{(g)}}{\sigma^{(g)}}$ for $m = 1, \dots, \mu$
    $\mathbf{C}_\mu = \sum_{i=1}^{\mu} w_i \mathbf{z}_i {\mathbf{z}_i}^{\mathrm{T}}$
    $\mathbf{C}^{(g+1)} = (1 - \frac{1}{c_\tau})\mathbf{C}^{(g)} + \frac{1}{c_\tau}\mathbf{C}_\mu^{(g+1)}$
    $g = g + 1$
end while
```

Fig. 1. The main steps of a CMSA-ES. Normally, the weights w_m are set to $w_m = 1/\mu$ for $m = 1, \dots, \mu$.

3 Covariance Matrix Estimation: Introducing Shrinkage

Using the population covariance matrix necessitates an appropriate sample size with $\mu \gg N$ for obtaining a high quality estimator. If this is not the case, estimate and "true" covariance may not agree well, especially in the case of high-dimensional search spaces. Among others, the eigen structure may be significantly distorted, see e.g. [20]. However, in evolution strategies typical recommendations for the population sizing state to use an offspring population size λ of either $\lambda = \mathcal{O}(\log(N))$ or $\lambda = \mathcal{O}(N)$ and setting $\mu = \lceil c\lambda \rceil$ with $c \in (0, 0.5)$. Thus, either $\mu/N \to c$ or even $\mu/N \to 0$ for $N \to \infty$ holds which disagrees with the requirement.

As stated above, the estimation of covariance matrices has received a lot of attention in statistics, see e.g. [6,30,38] and several techniques have been introduced. This paper focuses on *linear shrinkage estimators* that can be computed comparatively efficiently. Other classes, as e.g. thresholding operators for sparse covariance matrix estimation, are currently considered in separate analyses [25,29]. Following [20,35], linear shrinkage approaches are based on

$$\mathbf{S}_{\text{est}}(\rho) = \rho \mathbf{F} + (1 - \rho)\mathbf{C}_\mu \tag{7}$$

with $\mathbf{F}$ the *target* to correct the estimate provided by the sample covariance $\mathbf{C}_\mu$. The parameter $\rho \in (0, 1)$ is called the *shrinkage intensity*. Equation (7) is used to shrink the eigenvalues of $\mathbf{C}_\mu$ towards the eigenvalues of $\mathbf{F}$. The intensity ρ should be chosen to minimize the expected error

$$\mathrm{E}\Big(\|\mathbf{S}_{\text{est}}(\rho) - \mathit{\Sigma}\|_F^2\Big) \tag{8}$$

with $\|\cdot\|_F^2$ denoting the squared Frobenius norm given by

$$\|\mathbf{A}\|_F^2 = \frac{1}{N}\operatorname{Tr}\left[\mathbf{A}\mathbf{A}^{\mathrm{T}}\right], \tag{9}$$

see [20]. Note the factor $1/N$ is additionally introduced in [20] to normalize the norm w.r.t. the dimension.

Based on (8) and taking into account that the true covariance is unknown in practice, Ledoit and Wolf were able to obtain an optimal shrinkage intensity for the target $\mathbf{F} = \operatorname{Tr}(\mathbf{C}_\mu)/N\,\mathbf{I}$ for general probability distributions.

Several other approaches can be identified in literature. On the one hand, different targets can be considered, see e.g. [11,18,34,39]. Schäfer and Strimmer analyzed among others diagonal matrices with equal and unequal variance or special correlation models [34]. Fisher and Sun also allowed for several targets [11] assuming a multivariate normal distribution. Toulumis relaxed the normality assumption, considered several targets, and provided a new non-parametric family of shrinkage estimators [39]. Other authors introduced different estimators, see e.g. [7] or [6]). Recently, Ledoit and Wolf extended their work to include non-linear shrinkage estimators [21,22].

A problem arises concerning the complexity of the approaches. The associated optimization problem has to be solved which may be a task of its own especially in the case of non-linear estimates. Since the estimation has to be performed in every generation of the ES, only computationally simple approaches can be taken into account. Therefore, the paper focuses on linear shrinkage with shrinkage targets and intensities taken from [11,19,20,39].

Furthermore, transferring shrinkage estimators to ESs needs to consider the situation in which the estimation occurs since it differs from the assumptions in statistical literature:

- First, the covariance matrix $\Sigma = \mathbf{C}^{(g)}$ that was used to create the offspring is known.
- Second, the sample is based on truncation selection. Therefore, the variables cannot be assumed to be independent and identically distributed (iid). However, this is one of the main assumptions for deriving alternative estimators for the covariance. In the case of evolution strategies, the sample $\mathbf{x}_{1:\lambda}, \ldots, \mathbf{x}_{\mu:\lambda}$ would only represent normally distributed random variables if there were no selection pressure.

In this context, it is interesting to note that in the discussion [12] with respect to the setting of the CMA-ES control parameters it is argued to choose the values so that the distribution of the random variables would remain the same if there were no selection effects. This paper uses a similar argument to justify the usage of the shrinkage intensities obtained for assuming iid random or even normally distributed random variables. Since we are aware of the fact that the situation may differ considerably from the prerequirements in the statistical literature, other settings are also taken into account.

Before continuing, a closer look at the covariance matrix update (4) may be interesting. Equation (4) of the ES algorithm represents a special case of shrinkage with the old covariance matrix as the target. The shrinkage intensity is determined with $c_\tau = 1 + [N(N+1)]/[2\mu]$, Eq. (5), as $\rho = 1 - 1/c_\tau$. If $\mu \gg N$, $\rho \approx 0$ holds and the sample covariance is only slightly corrected. Otherwise, if $N \gg \mu$, the old covariance gains importance. To illustrate the main effects of (4), the eigenspace of $\mathbf{C}^{(g)}$ is considered. Since the covariance matrix is a positive definite matrix, a spectral composition of $\mathbf{C}^{(g)}$ with $\mathbf{C}^{(g)} = \mathbf{M}^{\mathrm{T}} \Lambda \mathbf{M}$ can be conducted. The modal matrix $\mathbf{M} = (\mathbf{v}_1, \ldots, \mathbf{v}_N)$ contains the eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_N$ of $\mathbf{C}^{(g)}$, whereas $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ represents a diagonal matrix with the corresponding eigenvalues $\lambda_1, \ldots, \lambda_N$. The representation $\mathbf{C}_C^{(g+1)}$ of $\mathbf{C}^{(g+1)}$ in the eigenspace of $\mathbf{C}^{(g)}$ then reads

$$\begin{aligned} \mathbf{C}_C^{(g+1)} &= \rho \Lambda + (1-\rho)\mathbf{C}_\mu^C \\ &= \mathrm{diag}(\mathbf{C}_\mu^C) + \rho\Big(\Lambda - \mathrm{diag}(\mathbf{C}_\mu^C)\Big) + (1-\rho)\Big(\mathbf{L}_\mu^C + \mathbf{U}_\mu^C\Big) \end{aligned} \tag{10}$$

with $\mathbf{C}_\mu^C = \mathbf{M}^{\mathrm{T}} \mathbf{C}_\mu \mathbf{M}$. The matrix $\mathbf{L}_\mu^C$ denotes the matrix with the entries of $\mathbf{C}_\mu^C$ below the diagonal, whereas $\mathbf{U}_\mu^C$ comprises the elements above. As Eq. (10) shows, the covariance matrix update (4) decreases the off-diagonal elements of the population covariance in the eigenspace. In the case of the diagonal entries, two cases may appear: if $c_{\mu_{ii}}^C < \lambda_i$, the new entry is in the interval $[c_{\mu_{ii}}^C, \lambda_i]$ and thus the estimate increases towards λ_i, otherwise it is shrunk towards λ_i. Thus, in the eigenspace of $\mathbf{C}^{(g)}$, Eq. (10) behaves similarly to shrinkage with a diagonal matrix as target matrix. Therefore, in original space, it shrinks the eigenvalues of the population matrix towards those of the target. In contrast to shrinkage, the target matrix is the old covariance (which is not obtainable in the general case). If the sample were drawn from independent and identically distributed variables, the update would "correct" the distortion due to the small sample size with the previous and in that case also the true covariance. Considering that a shrinkage procedure is already present in the original CMSA-ES, the question naturally arises, whether the strategy may benefit from additional corrections of the sample covariance.

Applying shrinkage requires among others the choice of an appropriate target. Most approaches consider regular structures as e.g. the scaled unity matrix, diagonal matrices, or matrices with constant correlations. However, a shrinkage towards a regular structure w.r.t. the coordinate system of the original search space does not appear as a good choice concerning the optimization of arbitrary functions.

4 Evolution Strategies and Shrinkage Estimators

Since we cannot assume that the covariance matrix adaptation would profit from correcting the estimate towards regular structures in every application case, shrinkage in the original search space is not taken into account. Instead, appropriate space transformations are investigated. The resulting ES algorithms will follow the same general principle:

1. A suitable transformation of the coordinate system is conducted.
2. A shrinkage estimation is performed in the transformed space.
3. After re-transformation to the original space, the result is used for the covariance matrix update.

As illustrated in the previous section, the original CMSA-update (4) shrinks the sample covariance towards a diagonal matrix in the eigenspace of the previous covariance matrix $\mathbf{C}^{(g)}$. Therefore, this eigenspace will also be taken into account for the current investigation.

Space transformations have been considered in ESs before. For example, Hansen argued in [14] that changing the coordinate system may improve the performance. For this reason, he introduced an adaptive encoding for the CMA-ES. It is based on a spectral decomposition of the covariance matrix. New search points are created in the eigenspace of the covariance matrix. Similar to [14], we assume that the ES may benefit from a change of the coordinate system. However, the covariance matrix adaptation and estimation which in [14] occur in the original space will be carried out in the transformed space.

Furthermore, other spaces may be also be beneficial. For example, [37] introduced an additional potential transformation. In [37], the authors were faced with the task to obtain a reliable covariance matrix. To this end, a sample covariance matrix $\mathbf{S}_i$ was combined with a pooled variance matrix $\mathbf{C}_p$ – similar to (4)

$$\mathbf{S}_{mix}(\xi) = \xi\mathbf{C}_p + (1-\xi)\mathbf{S}_i \tag{11}$$

with the parameter ξ to be determined. To continue, the authors switched to the eigenspace of the non-weighted mixture matrix where they followed a maximal entropy approach to determine an improved estimate of the covariance matrix. Based on [14,37], this paper considers the following choices for the transformation matrix which arise as combinations of the population covariance matrix $\mathbf{C}_\mu$ and $\mathbf{C}^{(g)}$

$$\mathbf{S}_{mix} = \mathbf{C}^{(g)} + \mathbf{C}_\mu, \tag{12}$$

$$\mathbf{S}_{(g+1)} = (1-c_\tau)\mathbf{C}^{(g)} + c_\tau\mathbf{C}_\mu, \tag{13}$$

$$\mathbf{S}_{(g)} = \mathbf{C}^{(g)}. \tag{14}$$

The variants (12)–(14) are based on different assumptions: The first (12) follows [37]. The influence of the old covariance and the population covariance are balanced. Structural changes caused by $\mathbf{C}_\mu$ will be dampened but will influence the result more strongly than in the case of (13) and (14). The second (14) uses the covariance mixture that appears in the original CMSA-ES. Depending on the size of c_τ, which is in turn a function of μ and N, see (5), the influence of the population covariance matrix may be stronger or lesser. The third considers the eigenspace of the old covariance matrix and reduces therefore the influence of the new estimate. Equations (12)–(14) are used to change the coordinate system. The representations of the covariance matrices in the eigenspace are given as $\mathbf{C}_\mu^S := \mathbf{M}_S^{\mathrm{T}}\mathbf{C}_\mu\mathbf{M}_S$ and $\mathbf{C}_S := \mathbf{M}_S^{\mathrm{T}}\mathbf{C}^{(g)}\mathbf{M}_S$ with S standing for one of the variants (12)–(14). Assuming that for example (14) is used, the following steps are performed

- spectral decomposition: $\mathbf{M}, \mathbf{D} \leftarrow \text{spectral}(\mathbf{S}_{(g)})$,
- determination of $\mathbf{C}_\mu^S := \mathbf{M}_S^{\mathrm{T}} \mathbf{C}_\mu \mathbf{M}_S$ and $\mathbf{C}_S := \mathbf{M}_S^{\mathrm{T}} \mathbf{C}^{(g)} \mathbf{M}_S$,
- shrinkage estimation resulting in $\hat{\mathbf{C}}_{\text{shr}}$,
- retransformation $\mathbf{C}_\mu = \mathbf{M}^{\mathrm{T}} \hat{\mathbf{C}}_{\text{shr}} \mathbf{M}$,
- covariance adaptation

$$\mathbf{C}^{(g+1)} = (1 - \frac{1}{c_\tau}) \mathbf{C}^{(g)} + \frac{1}{c_\tau} \mathbf{C}_\mu.$$

They substitute Line 13 of the CMSA-ES algorithm in Fig. 1. Once the space transformation has been conducted, different targets can be taken into account. The analysis considers the matrices

$$\mathbf{F}_{\mathrm{u}} = v\mathbf{I}, \tag{15}$$

with $v = \mathrm{Tr}(\mathbf{C}_\mu^S)/N$ [20],

$$\mathbf{F}_{\mathrm{d}} = \mathrm{diag}(\mathbf{C}_\mu^S) \tag{16}$$

the diagonal entries of $\mathbf{C}_\mu^S$ [11,39], the constant correlation model with matrix $\mathbf{F}_{\mathrm{c}}$ the entries of which read

$$f_{ij} = \begin{cases} s_{ii} & \text{if } i = j \\ \bar{r}\sqrt{s_{ii}s_{jj}} & \text{if } i \neq j \end{cases} \tag{17}$$

and $\bar{r} = 2/((N-1)N) \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} s_{ij}/\sqrt{s_{ii}s_{jj}}$ [19]. The shrinkage intensities are taken from the corresponding publications. For (15) the parameter is based on [20], for (16) it follows [11,39] while it is taken from [19] in the case of (17).

Since the new ES uses an explicit shrinkage, the question arises whether the additional term consisting of the old covariance matrix in (4) remains necessary or whether the ES may operate solely with shrinkage. Preliminary investigations indicate that the latter strategies perform worse than the original CMSA-ES. Therefore, the current analysis only considers a combination of both. It should also be noted that similar to the original update, the previous covariance could be used to determine the target instead of the sample. Both cases will be investigated more closely in future research.

5 Comparing Shrinkage Approaches: An Experimental Analysis

Experiments were conducted to investigate the shrinkage estimators introduced. First, the question of finding a suitable transformation was addressed. To this end, a comparison of the effects of (12)–(14) was carried out for a combination of (16) and the shrinkage intensity taken from [39]. Preliminary experiments showed that (14) lead to good results. Therefore, the remaining discussion in this paper is restricted to ESs using a transformation with the old covariance matrix. However, the increased variability provided by (12) and (13) should be considered together with (17) or (15) in further experiments.

The analysis considers ES-algorithms which apply shrinkage estimators as defined in (15) to (17). Aside from the CMSA-ES, we denote the strategies as follows

1. CI-ES: a CMSA-ES using (15) as shrinkage target,
2. CC-ES: the CMSA-ES with the constant correlation model (17),
3. FS-ES: the CMSA-ES which uses (16) and follows [11] to determine the shrinkage intensity,
4. Tou-ES: a CMSA-ES based on (16) which uses the shrinkage intensity of [39].

The approaches were coded in MATLAB. In the case of the CI-ES and the C-ES, we used the estimation source code provided by the authors on their webpage[1]. The implementation of the Tou-ES follows closely the R package[2]. In the analysis presented here, we did not use the oracle shrinkage intensity for the FS-ES as in [26] but applied the optimal estimated value (11) in [11, p. 1913]. Therefore, the results may differ from [26]. This ES also operates with a maximal number of fitness evaluations of $\mathrm{FE}_{\max} = 3 \times 10^5 N$.

5.1 Experimental Set-Up

The parameters for the experiments read as follows. Each experiment uses 15 repeats. The initial population is drawn uniformly from $[-4, 4]^N$, whereas the mutation strength is chosen from $[0.25, 1]$. The search space dimensions were set to 10 and 20. The maximal number of fitness evaluations is given by $\mathrm{FE}_{\max} = 2 \times 10^5 N$. All evolution strategies use $\lambda = \lfloor \log(3N) + 8 \rfloor$ offspring and $\mu = \lceil \lambda/4 \rceil$ parents. A run terminates prematurely if the difference between the best value so far and the optimal fitness value $|f_{\text{best}} - f_{\text{opt}}|$ is below a predefined precision set to 10^{-8}. Furthermore, we introduce a restart mechanism into the ESs so that the search is re-initialised when the search has stagnated for $10 + \lceil 30N/\lambda \rceil$ generations. Stagnation is determined by measuring the best function values in a generation. If the difference between minimal and maximal values of the sample lies below 10^{-8} for the given time-interval, the ES does not make significant movements anymore and the search is started anew.

The experiments are conducted with the help of the black box optimization benchmarking (BBOB) software framework[3] and the test suite, see [13]. The framework allows the analysis of algorithms and provides means to generate tables and figures of the results together with the corresponding legends.

---

[1] http://www.econ.uzh.ch/faculty/wolf/publications.html.

[2] http://cran.r-project.org/web/packages/ShrinkCovMat.

[3] The latest software can be obtained from COmparing Continuous Optimisers: COCO (http://coco.gforge.inria.fr/).

This paper considers 24 noise-less functions [9]. They consist of four subgroups: separable functions (function ids 1–5), functions with low/moderate conditioning (ids 6–9), functions with high conditioning (ids 10–14), and two groups of multimodal functions (ids 15–24).

The experiments use the expected running time (ERT) as performance measure. The ERT is defined as the expected value of the function evaluations (f-evaluations) the algorithm needs to reach the target value with the required precision for the first time, see [13]. In this paper, the estimate

$$\mathrm{ERT} = \frac{\#(FEs(f_{\text{best}} \geq f_{\text{target}}))}{\#succ} \tag{18}$$

is used, that is, the fitness evaluations $FEs(f_{\text{best}} \geq f_{\text{target}})$ of each run until the fitness of the best individual is smaller than the target value are summed up and divided by the number of successful runs.

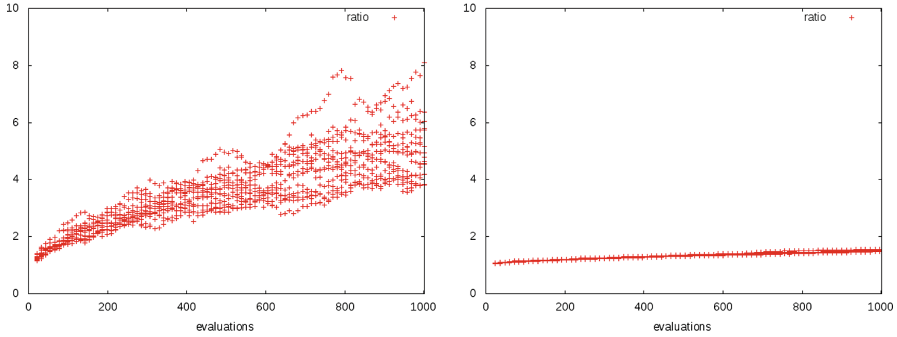
5.2 Results and Discussion

First of all, the behavior of the strategies for two exemplary functions, the sphere, $f(\mathbf{x}) = \|\mathbf{x}\|^2$, and the discus, $f(\mathbf{x}) = 10^6 x_1^2 + \sum_{i=2}^{N} x_i^2$ is investigated. The functions were selected since they represent very different optimization tasks. Figures 2 and 3 show the ratio of the largest to the smallest eigenvalue of the covariance matrix for the CMSA-ES and for two of the shrinkage approaches, the CC-ES and the FS-ES. An ES optimizing the sphere should keep the ratio close to one, whereas the ratio should increase for the discus. In the case of the sphere, the figures illustrate that the largest and smallest eigenvalue develop differently and diverge for $N = 10$. Shrinkage causes the problem to be less pronounced.

In the case of the discus, different eigenvalues are expected. All strategies achieve this, the ESs with shrinkage operators show again a lower rate of increase. This may be a hint that the adaptation process of the covariance matrix may be decelerated by the additional shrinkage. Figure 3 shows an even slower increase of the ratio for the FS-ES than for the CC-ES. Whether this lowers the performance is investigated in the second series of experiments. Figure 4 shows the empirical cumulative distribution functions for $N = 10$ and $N = 20$. Whether introducing shrinkage terms improves the performance of the ES depends on the function class. In the case of ill-conditioned functions, shrinkage targets consisting of the diagonal elements of the transformed sample covariance may offer benefits.

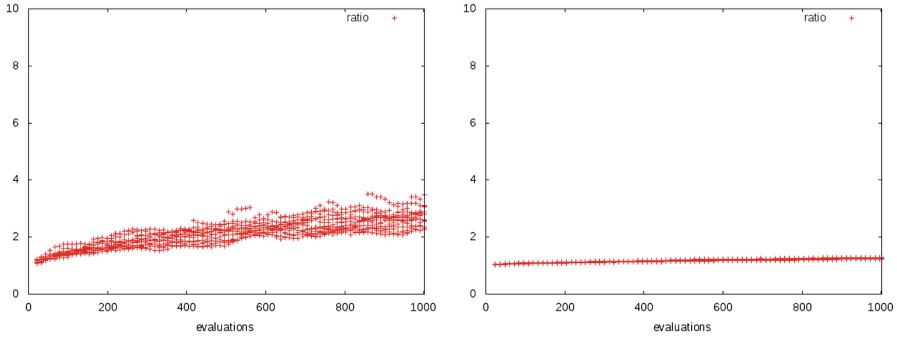
The results of the experiments can be examined more closely with the help of Table 1 ($N = 10$) and Table 2 ($N = 20$). They provide the estimate of the expected running time (ERT) for several precision targets ranging from 10^1 to 10^{-7}. Also shown is the number of successful runs.

Several functions represent challenges for the ESs considered. These comprise the Rastrigin functions (id 3, id 4, id 15, and id 24) of the test suite, the step ellipsoidal function with a condition number of 100 (id 7), and a multi-modal function with a weak global structure based on Schwefel (id 20). In these cases, all strategies are unable to progress further than the first intermediate precision



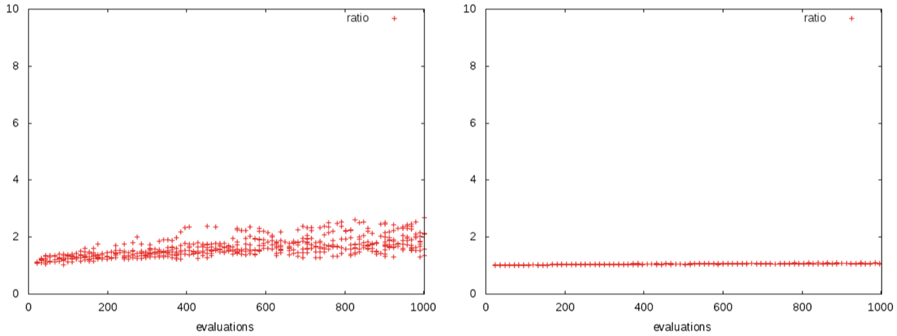
(a) CMSA-ES, sphere, $N = 10$

(b) CMSA-ES, sphere, $N = 40$



(c) CC-ES, sphere, $N = 10$

(d) CC-ES, sphere, $N = 40$



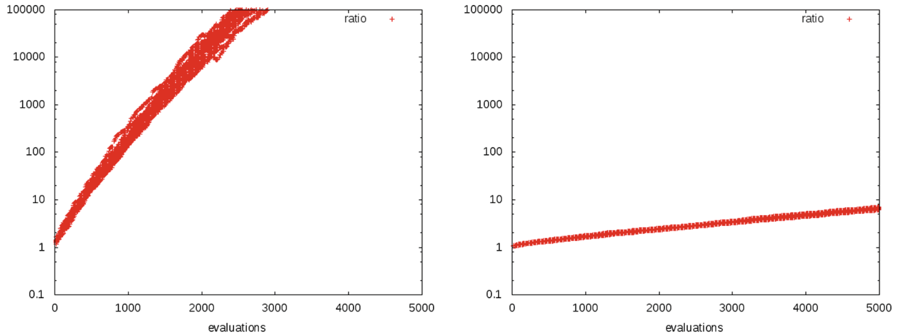
(e) FS-ES, sphere, $N = 10$

(f) FS-ES, sphere, $N = 40$

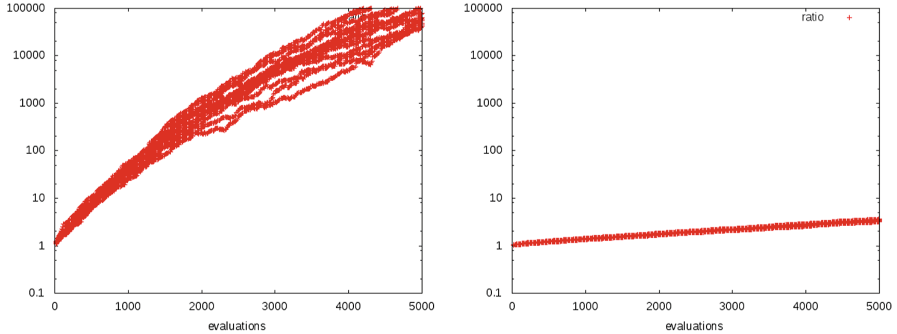
Fig. 2. The development of the ratio of the largest to the smallest eigenvalue of the covariance on the sphere for the CMSA-ES, the CC-ES, and the FS-ES. Shown are the results from 15 runs per dimensionality.

of 10^1. Additionally, for the multi-modal functions, 19 (composite Griewank-Rosenbrock function) and 23 (Katsuura), no strategy could reach 10^{-1}. The functions in question are therefore removed from the tables.

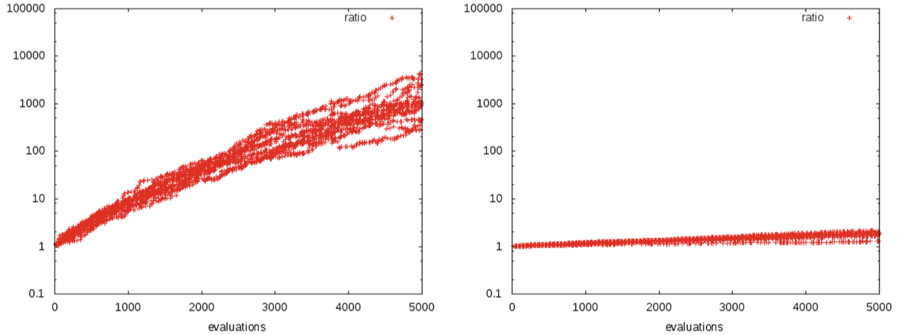
To analyze the remaining functions, the four groups of the test suite are taken into account. The first class comprises the separable functions with id 1 to id 5.



(a) CMSA-ES, discus, $N = 10$ (b) CMSA-ES, discus, $N = 40$



(c) CC-ES, discus, $N = 10$ (d) CC-ES, discus, $N = 40$



(e) FS-ES, discus, $N = 10$ (f) FS-ES, discus, $N = 40$

Fig. 3. The development of the ratio of the largest to the smallest eigenvalue of the covariance on the discus. Shown are the results from 15 runs per dimensionality.

The three remaining functions, the sphere (f1), the separable ellipsoidal function (f2), and the linear slope (f5) differ in the degree of difficulty for the strategies. All strategies do not show any problems on the sphere or on the slope. Here, several shrinkage variants perform similarly to or sometimes may even surpass the original version. However, the differences are not statistically significant.

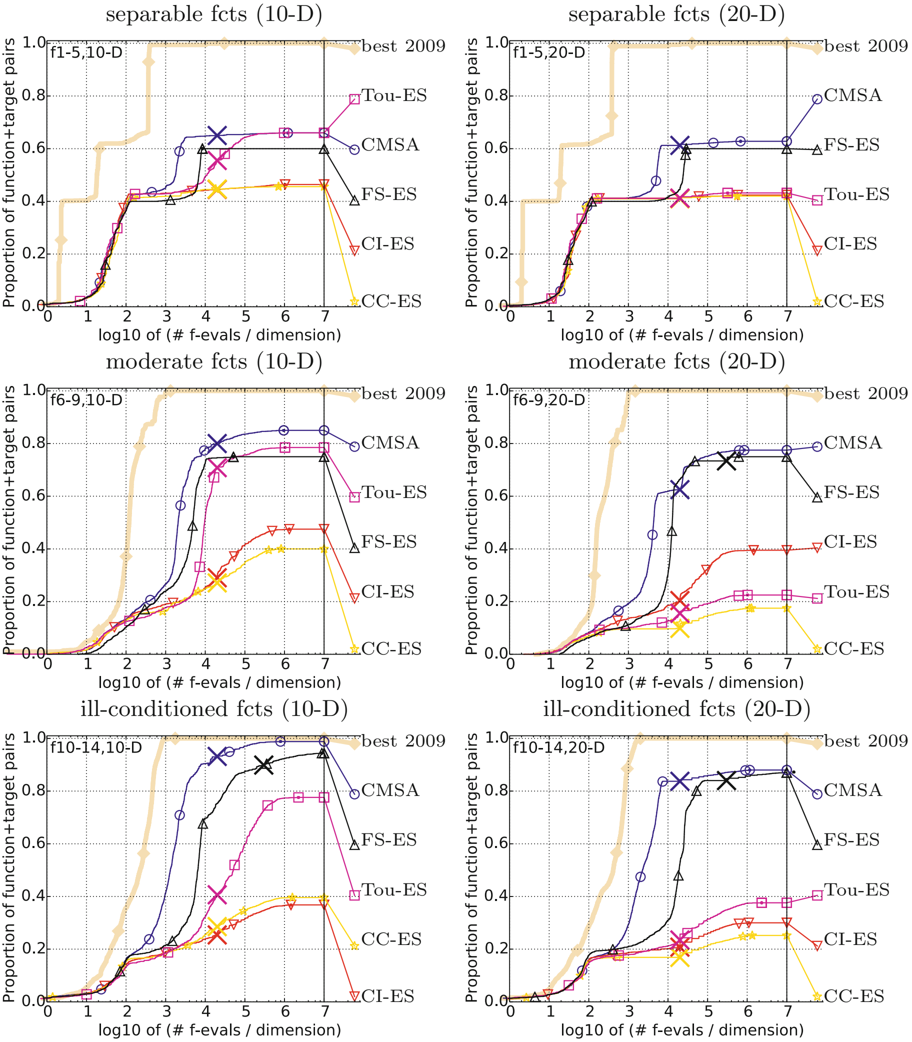


Fig. 4. Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 50 targets in $10^{[-8..2]}$ for 10-D and 20-D.

The more difficult ellipsoidal function, however, cannot be solved by the CC-ES, the CI-ES, and for $N = 20$ by the Tou-ES. Interestingly, the FS-ES achieves successful runs for all search space dimensionalities. Its performance is significantly worse than the original CMSA-ES, however. Two effects may play a role. The ellipsoidal function, defined by $f(\mathbf{x}) = \sum_{i=1}^{N} 10^{6(i-1)/(N-1)} x_i^2$, is not solved well by ESs with covariance matrices treating all directions with the same weight. The matrix used in the transformation may not be sufficient to provide the variability required when combined with restrictive structures. The target matrices supplied by the CI-ES may be too regular for the ES to be able to adapt with the necessary velocity. Why the CC-ES also exhibits problems, although it represents a more general model, merits further investigations. The failure of the Tou-ES to achieve the final precision target may be due to the shrinkage intensity since this is the only point where it differs from the FS-ES. For this reason, Sect. 6.2 provides a first analysis of the impact of this parameter on the performance of the ES. In future research, the interaction with the parameter c_τ will be investigated more closely. Since this parameter approaches infinity for the typical μ/N ratios and increasing dimensionalities, the influence of the sample covariance lessens. Regularizing the covariance matrix may therefore be more important for smaller to medium search space dimensionalities. Concerning the question whether shrinkage improves the performance, no clear answer can be provided for the group of separable functions since the ellipsoidal function apparently requires a faster adaptation than the current versions supply.

The second group of functions consists of the attractive sector function (id 6), the step ellipsoidal function (id 7, results not shown), the original Rosenbrock function (id 8), and a rotated Rosenbrock function (id 9). These functions have low to moderate conditioning. The sector function is difficult to solve for all strategies. For $N = 20$, successful runs were recorded for the CMSA-ES and the CI-ES but only for a few cases, i.e., two or three. Therefore, the question arises whether initialization effects may have played a role in other words whether the ES in these cases started in the vicinity of the optimal point. For $N = 10$ and $N = 20$, the FS-ES reaches the final target precision of 10^{-8} in 15 of 15 runs, a result not mirrored by the other strategies. Concerning the Rosenbrock functions (f8 and f9), the CMSA-ES and the FS-ES perform best with the CMSA-ES being superior.

Functions with high conditioning constitute the next group. For the ellipsoidal function (f10), the discus (f11), the bent cigar (f12), the sharp ridge (f13), and the different powers function (f14), the CMSA-ES appears as the best performing strategy, followed by the FS-ES and the Tou-ES. More experiments are required since the bent cigar apparently represents a stronger challenge for the FS-ES for $N = 10$ than for $N = 20$.

In the case of the multi-modal functions (ids 15–24), all ESs encounter problems. Only for the two Gallagher's problems (id 21 and id 22) successes are recorded. Here, the CMSA-ES and the versions that use the diagonal elements of the sample covariance as shrinkage target show the best results. The FS-ES appears to be a good choice for $N = 10$ for function 21 whereas the CMSA-ES and the Tou-ES require fewer function evaluations for 22. For $N = 20$, only a few runs of all ESs reach the final target precision. Therefore, a comparison for the higher-dimensional search spaces is difficult and is not carried out in this paper.

Table 1. Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 in dimension 10. The ERT and in braces, as dispersion measure, the half difference between 90 and 10%-tile of bootstrapped run lengths appear for each algorithm and target, the corresponding best ERT in the first row. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{\mathrm{opt}} + 10^{-}8$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0{:}05$ or $p = 10^{-}k$ when the number k following the star is larger than 1, with Bonferroni correction by the number of instances.

10-D

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	22	23	23	23	23	23	23	15/15
CMSA	4.0(1)	8.6(3)	14(3)	19(5)	26(7)	38(5)	50(7)	15/15
CC-ES	4.2(2)	9.3(5)	14(3)	19(4)	24(5)	**34**(6)	**43**(2)	15/15
CI-ES	3.7(1)	8.4(2)	13(3)	**18**(4)	**23**(2)	35(4)	45(6)	15/15
Tou-ES	4.2(1)	9.1(3)	15(5)	21(4)	27(5)	40(6)	54(8)	15/15
FS-ES	**3.6**(2)	**7.8**(3)	**13**(3)	19(4)	25(4)	36(4)	47(4)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2	187	190	191	191	193	194	195	15/15
CMSA	**65**(27)$^{\star 4}$	**85**(23)$^{\star 4}$	**96**(20)$^{\star 4}$	**105**(19)$^{\star 4}$	**109**(10)$^{\star 4}$	**113**(19)$^{\star 4}$	**129**(63)$^{\star 4}$	15/15
CC-ES	∞	∞	∞	∞	∞	∞	∞ *2e5*	0/15
CI-ES	∞	∞	∞	∞	∞	∞	∞ *2e5*	0/15
Tou-ES	496(794)	850(462)	1517(1236)	1835(1008)	1833(1854)	2248(2160)	2757(929)	5/15
FS-ES	232(132)	306(96)	342(41)	358(50)	366(31)	376(17)	383(37)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f5	20	20	20	20	20	20	20	15/15
CMSA	**12**(5)	**17**(8)	**17**(7)	**17**(7)	**17**(8)	**17**(8)	**17**(8)	15/15
CC-ES	14(9)	19(9)	20(7)	20(8)	20(16)	20(9)	20(15)	15/15
CI-ES	18(15)	102(14)	104(10)	104(16)	104(306)	104(306)	104(9)	15/15
Tou-ES	12(6)	18(6)	19(31)	19(30)	19(16)	19(22)	19(10)	15/15
FS-ES	13(6)	18(5)	19(6)	19(6)	19(9)	19(6)	19(5)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f11	266	1041	2602	2954	3338	4092	4843	15/15
CMSA	**14**(3)$^{\star 4}$	**6.2**(3)$^{\star 4}$	**3.2**(1)$^{\star 4}$	**3.4**(0.7)$^{\star 4}$	**3.5**(2)$^{\star 4}$	**3.3**(2)$^{\star 4}$	**3.0**(2)$^{\star 4}$	15/15
CC-ES	∞	∞	∞	∞	∞	∞	∞ *2e5*	0/15
CI-ES	1.1e4(8256)	∞	∞	∞	∞	∞	∞ *2e5*	0/15
Tou-ES	152(266)	169(59)	150(181)	312(242)	277(374)	343(376)	596(475)	1/15
FS-ES	93(55)	42(14)	20(2)	19(4)	17(4)	15(2)	13(1)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f12	515	896	1240	1390	1569	3660	5154	15/15
CMSA	**4.2**(0.2)	**10**(21)	**13**(7)	**15**(16)$^{\star 2}$	**16**(15)$^{\star 4}$	**8.8**(7)$^{\star 4}$	**7.8**(2)$^{\star 4}$	15/15
CC-ES	29(0.3)	615(725)	2258(2136)	2020(2518)	∞	∞	∞ *2e5*	0/15
CI-ES	11(17)	37(48)	68(110)	161(315)	572(449)	∞	∞ *2e5*	0/15
Tou-ES	76(325)	177(188)	722(1392)	1016(1120)	910(1473)	799(984)	569(766)	1/15
FS-ES	5.8(30)	110(101)	167(84)	191(99)	203(65)	121(30)	∞ *3e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f13	387	596	797	1014	4587	6208	7779	15/15
CMSA	15(8)	**19**(14)	**31**(30)	**58**(131)	**28**(27)	**89**(61)	**185**(219)	2/15
CC-ES	32(4)	57(146)	87(157)	229(222)	145(194)	∞	∞ *2e5*	0/15
CI-ES	19(67)	59(93)	186(268)	610(961)	637(752)	∞	∞ *2e5*	0/15
Tou-ES	22(18)	50(65)	117(133)	500(750)	307(357)	∞	∞ *2e5*	0/15
FS-ES	**6.7**(10)	31(34)	73(90)	167(218)	116(149)	637(634)	∞ *3e6*	0/15

(Continued)

Table 1. *(Continued)*

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f6	412	623	826	1039	1292	1841	2370	15/15
CMSA	1.4(0.6)	3.3(6)	11(16)	14(21)	19(14)	25(23)	163(178)	6/15
CC-ES	37(2)	60(87)	183(276)	219(262)	348(346)	452(592)	1206(1139)	1/15
CI-ES	**1.2**(0.3)	18(49)	68(132)	193(200)	1088(1159)	∞	∞ *2e5*	0/15
Tou-ES	5.3(0.3)	89(65)	135(132)	138(92)	151(179)	369(518)	∞ *2e5*	0/15
FS-ES	1.3(0.6)	**3.3**(1)	**5.5**(6)	**7.5**(7)	**11**(7)	**17**(19)	**28**(61)*	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f8	326	921	1114	1217	1267	1315	1343	15/15
CMSA	**3.3**(5)	**17**(13)	**18**(9)*	**18**(3)$^{\star 2}$	**18**(9)$^{\star 2}$	**19**(5)$^{\star 3}$	**19**(9)$^{\star 3}$	15/15
CC-ES	75(316)	197(115)	∞	∞	∞	∞	∞ *2e5*	0/15
CI-ES	17(1)	378(279)	784(1350)	2310(4025)	∞	∞	∞ *2e5*	0/15
Tou-ES	11(12)	68(57)	72(43)	76(37)	77(47)	78(33)	79(35)	15/15
FS-ES	5.5(2)	37(35)	40(24)	42(13)	43(23)	45(28)	46(17)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f9	200	648	857	993	1065	1138	1185	15/15
CMSA	2.3(0.8)	**25**(17)	**24**(8)*	**22**(12)$^{\star 2}$	**22**(10)$^{\star 3}$	**21**(6)$^{\star 3}$	**21**(3)$^{\star 3}$	15/15
CC-ES	**2.2**(1.0)	323(223)	∞	∞	∞	∞	∞ *2e5*	0/15
CI-ES	16(36)	2122(2236)	∞	∞	∞	∞	∞ *2e5*	0/15
Tou-ES	4.8(9)	91(64)	86(47)	86(30)	84(24)	84(22)	83(18)	15/15
FS-ES	5.1(6)	37(20)	42(15)	43(9)	43(14)	44(7)	45(20)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f10	1835	2172	2455	2728	2802	4543	4739	15/15
CMSA	**6.5**(2)$^{\star 4}$	**7.6**(4)$^{\star 4}$	**7.7**(3)$^{\star 4}$	**7.5**(3)$^{\star 4}$	**7.6**(3)$^{\star 4}$	**4.9**(2)$^{\star 4}$	**4.9**(2)$^{\star 4}$	15/15
CC-ES	∞	∞	∞	∞	∞	∞	∞ *2e5*	0/15
CI-ES	∞	∞	∞	∞	∞	∞	∞ *2e5*	0/15
Tou-ES	111(225)	301(332)	608(652)	∞	∞	∞	∞ *2e5*	0/15
FS-ES	27(14)	28(11)	29(5)	27(4)	27(4)	17(3)	17(1)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f14	37	98	133	205	392	687	4305	15/15
CMSA	1.1(0.7)	**2.2**(0.9)	**2.8**(0.7)	**3.5**(1)	**4.3**(1)$^{\star 2}$	**8.7**(2)$^{\star 4}$	**4.8**(3)$^{\star 4}$	15/15
CC-ES	1.2(0.9)	2.4(0.8)	2.9(0.8)	3.6(0.8)	24(16)	∞	∞ *2e5*	0/15
CI-ES	**0.98**(0.6)	2.4(1)	2.9(2)	3.6(1)	72(42)	∞	∞ *2e5*	0/15
Tou-ES	1.4(1)	2.7(1)	3.5(2)	5.4(4)	24(13)	241(146)	339(453)	2/15
FS-ES	1.2(0.9)	2.6(1)	3.1(0.4)	3.8(0.8)	10(3)	41(10)	73(66)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f18	238	836	7012	15928	27536	37234	42708	15/15
CMSA	68(234)	129(186)	124(121)	∞	∞	∞	∞ *2e5*	0/15
CC-ES	3.3(17)	136(239)	124(171)	∞	∞	∞	∞ *2e5*	0/15
CI-ES	8.8(0.5)	**38**(86)	**55**(22)	∞	∞	∞	∞ *2e5*	0/15
Tou-ES	8.5(0.6)	187(239)	211(257)	∞	∞	∞	∞ *2e5*	0/15
FS-ES	**1.3**(1)	342(575)	334(408)	**2822**(3673)	∞	∞	∞ *3e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f21	130	2236	4392	4487	4618	5074	11329	8/15
CMSA	**9.5**(28)	23(70)	20(48)	20(22)	19(25)	17(38)	7.8(11)	12/15
CC-ES	47(235)	109(143)	58(67)	57(77)	55(47)	50(49)	23(47)	8/15
CI-ES	33(57)	18(12)	13(10)	17(23)	16(23)	36(63)	35(43)	6/15
Tou-ES	50(39)	39(134)	32(48)	31(34)	30(24)	28(20)	14(17)	9/15
FS-ES	13(12)	**14**(23)	**8.8**(10)	**8.7**(8)	**8.5**(13)	**7.8**(9)	**3.5**(5)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f22	98	2839	6353	6620	6798	8296	10351	6/15
CMSA	**25**(45)	**6.3**(6)	**13**(27)	**12**(14)	**12**(13)	**10**(10)	**8.1**(28)	13/15
CC-ES	149(369)	29(73)	24(24)	25(31)	25(21)	30(32)	∞ *2e5*	0/15
CI-ES	48(118)	13(8)	23(32)	49(48)	133(187)	∞	∞ *2e5*	0/15
Tou-ES	61(51)	10(16)	17(36)	17(22)	17(25)	16(17)	14(9)	12/15
FS-ES	94(355)	8.8(10)	102(157)	99(186)	96(359)	79(120)	64(168)	13/15

Table 2. Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 in dimension 20. The ERT and in braces, as dispersion measure, the half difference between 90 and 10%-tile of bootstrapped run lengths appear for each algorithm and target, the corresponding best ERT in the first row. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{\text{opt}} + 10^{-8}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0{:}05$ or $p = 10^{-k}$ when the number k following the star is larger than 1, with Bonferroni correction by the number of instances.

20-D

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	43	43	43	43	43	43	43	15/15
CMSA	4.9(1)	10(2)	15(2)	19(2)	25(2)	34(2)	45(3)	15/15
CC-ES	5.0(1)	10(1)	15(3)	19(3)	24(2)	34(3)	44(3)	15/15
CI-ES	4.7(1)	**8.9**(1.0)	**14**(2)	**19**(3)	**23**(3)	**33**(4)	**42**(3)	15/15
Tou-ES	5.1(2)	10(2)	14(2)	19(3)	25(2)	35(3)	46(3)	15/15
FS-ES	**4.6**(0.6)	9.3(4)	14(3)	19(3)	24(3)	34(3)	44(4)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2	385	386	387	388	390	391	393	15/15
CMSA	**173**(38)$^{\star 4}$	**240**(32)$^{\star 4}$	**265**(27)$^{\star 4}$	**273**(25)$^{\star 4}$	**277**(44)$^{\star 4}$	**285**(42)$^{\star 4}$	**293**(24)$^{\star 4}$	15/15
CC-ES	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CI-ES	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
Tou-ES	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
FS-ES	666(283)	987(299)	1172(213)	1235(191)	1286(96)	1307(119)	1320(123)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f5	41	41	41	41	41	41	41	15/15
CMSA	12(5)	15(2)	15(6)	15(7)	15(3)	15(6)	15(6)	15/15
CC-ES	14(6)	18(14)	19(14)	19(7)	19(3)	19(15)	19(15)	15/15
CI-ES	13(5)	17(5)	18(7)	18(10)	18(7)	18(3)	18(7)	15/15
Tou-ES	**11**(3)	**14**(3)	**14**(6)	**14**(4)	**14**(6)	**14**(5)	**14**(5)	15/15
FS-ES	14(4)	17(15)	18(5)	18(6)	18(6)	18(15)	18(5)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f11	1002	2228	6278	8586	9762	12285	14831	15/15
CMSA	**12**(2)$^{\star 4}$	**7.5**(1)$^{\star 4}$	**3.1**(0.5)$^{\star 4}$	**2.6**(0.7)$^{\star 4}$	**2.6**(0.7)$^{\star 4}$	**2.5**(0.5)$^{\star 4}$	**2.5**(0.8)$^{\star 4}$	15/15
CC-ES	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CI-ES	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
Tou-ES	2831(2295)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
FS-ES	111(46)	108(41)	43(18)	35(10)	32(6)	28(2)	24(4)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f12	1042	1938	2740	3156	4140	12407	13827	15/15
CMSA	2.5(4)	**10**(9)	**13**(8)	**15**(8)	**14**(6)$^{\star 2}$	**5.9**(2)$^{\star 4}$	**6.2**(2)$^{\star 4}$	15/15
CC-ES	60(192)	182(361)	585(949)	1775(2599)	∞	∞	∞ *4e5*	0/15
CI-ES	97(192)	569(465)	2045(1606)	1775(2567)	∞	∞	∞ *4e5*	0/15
Tou-ES	97(96)	137(242)	315(657)	1775(1204)	∞	∞	∞ *4e5*	0/15
FS-ES	**1.4**(0.1)	49(110)	88(90)	123(126)	131(83)	74(26)	85(18)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f14	75	239	304	451	932	1648	15661	15/15
CMSA	1.8(0.7)	1.9(0.6)	2.5(0.3)	3.2(0.4)	**5.2**(0.6)$^{\star 4}$	**11**(4)$^{\star 4}$	**4.2**(1)$^{\star 4}$	15/15
CC-ES	1.8(1)	1.8(0.6)	2.4(0.4)	3.4(0.9)	23(7)	∞	∞ *4e5*	0/15
CI-ES	1.8(0.8)	1.9(0.8)	2.4(0.4)	3.1(0.6)	∞	∞	∞ *4e5*	0/15
Tou-ES	**1.5**(0.6)	**1.7**(0.5)	2.3(0.6)	3.5(0.7)	27(18)	∞	∞ *4e5*	0/15
FS-ES	2.0(0.8)	1.8(0.4)	**2.3**(0.4)	**3.0**(0.3)	11(2)	78(22)	32(6)	15/15

(continued)

Table 2. *(continued)*

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f6	1296	2343	3413	4255	5220	6728	8409	15/15
CMSA	**1.5**(1)	**2.5**(2)	**4.6**(3)	**12**(9)	**34**(51)	80(45)	331(595)	2/15
CC-ES	29(19)	88(172)	108(92)	154(283)	144(131)	253(236)	206(198)	3/15
CI-ES	80(234)	471(726)	1643(1319)	∞	∞	∞	∞ *4e5*	0/15
Tou-ES	56(10)	275(430)	∞	∞	∞	∞	∞ *4e5*	0/15
FS-ES	2.0(1.0)	7.6(9)	18(25)	29(30)	42(34)	**59**(38)	**71**(32)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f8	2039	3871	4040	4148	4219	4371	4484	15/15
CMSA	**11**(5)$^{\star 2}$	**30**(30)	**31**(8)*	**31**(78)*	**31**(5)*	**31**(28)*	**31**(46)*	13/15
CC-ES	106(61)	727(1144)	∞	∞	∞	∞	∞ *4e5*	0/15
CI-ES	2752(4562)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
Tou-ES	96(45)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
FS-ES	27(13)	39(8)	48(14)	51(13)	53(17)	54(11)	54(15)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f9	1716	3102	3277	3379	3455	3594	3727	15/15
CMSA	**17**(6)$^{\star 2}$	**40**(130)*	**41**(63)*	**41**(4)*	**41**(60)*	**40**(60)*	**40**(54)*	13/15
CC-ES	158(107)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CI-ES	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
Tou-ES	88(51)	633(773)	∞	∞	∞	∞	∞ *4e5*	0/15
FS-ES	38(6)	194(486)	196(4)	197(448)	195(440)	191(5)	186(6)	14/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f10	7413	8661	10735	13641	14920	17073	17476	15/15
CMSA	**10**(2)$^{\star 4}$	**11**(4)$^{\star 4}$	**9.2**(2)$^{\star 4}$	**7.8**(1)$^{\star 4}$	**7.3**(1)$^{\star 4}$	**6.7**(0.6)$^{\star 4}$	**6.8**(1)$^{\star 4}$	15/15
CC-ES	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
CI-ES	∞	∞	∞	∞	∞	∞	∞ *4e5*	0/15
Tou-ES	796(1389)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
FS-ES	36(12)	47(9)	44(3)	37(3)	34(3)	31(2)	31(1)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f16	1384	27265	77015	1.4e5	1.9e5	2.0e5	2.2e5	15/15
CMSA	22(72)	**95**(114)	∞	∞	∞	∞	∞ *4e5*	0/15
CC-ES	45(0.2)	95(88)	∞	∞	∞	∞	∞ *4e5*	0/15
CI-ES	45(72)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
Tou-ES	**21**(0.2)	205(143)	∞	∞	∞	∞	∞ *4e5*	0/15
FS-ES	668(2169)	1430(2146)	∞	∞	∞	∞	∞ *6e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f17	63	1030	4005	12242	30677	56288	80472	15/15
CMSA	**0.93**(0.9)	**196**(389)	∞	∞	∞	∞	∞ *4e5*	0/15
CC-ES	1.1(1.0)	445(1942)	∞	∞	∞	∞	∞ *4e5*	0/15
CI-ES	1.3(3)	341(97)	**650**(849)	∞	∞	∞	∞ *4e5*	0/15
Tou-ES	1.2(0.8)	778(1844)	1399(1398)	∞	∞	∞	∞ *4e5*	0/15
FS-ES	0.95(0.9)	6658(2e4)	9738(2e4)	**6862**(1e4)	∞	∞	∞ *6e6*	0/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f21	561	6541	14103	14318	14643	15567	17589	15/15
CMSA	179(356)	245(183)	113(149)	112(175)	109(82)	103(109)	91(91)	3/15
CC-ES	357(713)	398(275)	184(206)	182(168)	178(157)	167(122)	148(267)	2/15
CI-ES	357(534)	168(199)	113(128)	112(126)	109(75)	103(84)	91(119)	3/15
Tou-ES	**179**(356)	**122**(183)	**78**(92)	**77**(70)	**75**(96)	**71**(167)	**63**(131)	4/15
FS-ES	3887(8016)	5963(9403)	2765(3829)	2724(2095)	2663(3483)	2505(2794)	2217(2558)	2/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f22	467	5580	23491	24163	24948	26847	1.3e5	12/15
CMSA	572(1500)	287(108)	∞	∞	∞	∞	∞ *4e5*	0/15
CC-ES	980(1714)	466(699)	∞	∞	∞	∞	∞ *4e5*	0/15
CI-ES	**312**(429)	**108**(233)	∞	∞	∞	∞	∞ *4e5*	0/15
Tou-ES	572(1285)	287(556)	**239**(145)	**232**(393)	**225**(132)	**211**(283)	**43**(66)	1/15
FS-ES	1.5e4(2e4)	1.5e4(2e4)	∞	∞	∞	∞	∞ *6e6*	0/15

6 Further Experimental Analyses

The experiments revealed that at least in the eigenspace of the previous covariance matrix and for the intensities considered, the ES does not benefit at all from shrinkage towards more complex structures than offered by a diagonal matrix. This is surprising, since the model with constant correlations would allow more degrees of freedom for the evolutionary process. Therefore, further investigations – concerning especially the question of the choice of the shrinkage intensity – will be carried out. In addition, ESs with the scaled unity matrix as a target do not perform as well as strategies with different diagonal elements. This can possibly be traced back to the fact that this target is the most restrictive and impairs the general adaptability of the ES.

Therefore, the further discussion in this paper is limited to the shrinkage estimators that use the diagonal entries of the sample covariance as correction term. The two variants taken into account, the Fisher-Sun (FS-ES) estimator and the Toulumis technique (Tou-ES) operate with the same target, but the performance of the associated evolution strategies differs. Since the only difference between these two estimators lies in the shrinkage intensity, this section carries out an investigation of the dependency of the performance of the ES on the choice of the factor.

So far, the experiments were restricted to the noise-free case assuming the possibility of exact function evaluations. The effects of shrinkage estimators concerning noisy optimization remain to be taken into account. Since this represents an important research and application area of evolution strategies, exemplary experiments are carried out.

6.1 A Brief Investigation Concerning Noisy Optimization

This subsection addresses the question whether shrinkage estimators may be useful in the case of noisy optimization. *Noise* or more generally *uncertainty* is a common and important problem in practical optimization. Following [4,16], uncertainty comprises noise, robustness, dynamical changes, and function approximations. Many causes exist: Physical measurements may be necessary during the optimization which are usually imprecise to a certain degree. Situational changes may occur – for instance when trying to find the fastest route in a traffic network. For a closer explanation, let us reconsider the function $f : \mathbb{R}^N \rightarrow \mathbb{R}$ that is to be optimized. Noise means that the function evaluations are not exact and that disturbances, i.e., measurement errors or similar, need to be taken into account. These can be modelled by a random variable ϵ. Instead of the exact f-value at $\mathbf{x}$ only the noisy $\tilde{f}(\mathbf{x}) = g(f, \mathbf{x}, \epsilon)$ can be observed. Following the test suite [10], the multiplicative noise model

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x})\epsilon \tag{19}$$

is considered in the paper. In the experiments conducted for the sphere and the Rosenbrock function, the noise term ϵ is represented by a log-normally

Table 3. ERT and half-interquantile range (90%–10%) divided by the best ERT measured during BBOB 2009 for different Δf values in 10-D.

10-D						
Δf_{opt}	10	0.1	1e-3	1e-5	1e-7	#succ
$\mathbf{f_{101}}$	26	181	194	210	226	15/15
1:CMSA-ES	$3.0_{(2)}$	$1.7_{(0.2)}$	$2.9_{(0.4)}$	$3.9_{(0.4)}$	$4.9_{(0.8)}$	15/15
2:FS-ES	$2.8_{(1)}$	$1.6_{(0.4)}$	$2.8_{(0.2)}$	$3.6_{(0.5)}$	$4.5_{(0.3)}$	15/15
$\mathbf{f_{104}}$	610	16641	19364	20764	22011	15/15
1:CMSA-ES	$368\ _{(2)}$	$140_{(38)}$	∞	∞	$\infty\mathit{3.0e6}$	0 /15
2:FS-ES	$22_{(27)}$	$46_{(8)}$	$\mathbf{46}_{(79)}{}^{\star}$	$\mathbf{44}_{(4)}\ {}^{\star}$	$\mathbf{50}_{(78)}{}^{\star}$	13/15
$\mathbf{f_{107}}$	945	3871	7352	11340	14303	15/15
1:CMSA-ES	$1.5_{(2)}$	$365_{(213)}$	$224_{(3)}$	$147_{(4)}$	$117_{(4)}$	15/15
2:FS-ES	$0.92_{(0.3)}$	$\mathbf{2.1}_{(1)}{}^{\star 2}$	$\mathbf{2.7}_{(1)}{}^{\star 4}$	$\mathbf{2.8}_{(2)}{}^{\star 4}$	$\mathbf{2.9}_{(1)}{}^{\star 4}$	15/15
$\mathbf{f_{110}}$	11224	7.0e7	∞	∞	∞	0
1:CMSA-ES	$89_{(76)}$	$0.62_{(0.6)}$	∞	∞	∞	0/15
2:FS-ES	$2.5_{(0.9)}$	$0.10_{(0.1)}$	$\mathit{1.3e7}_{(1e7)}$	$\mathit{1.3e7}_{(1e7)}$	$\mathit{1.3e7}_{(3e7)}$	3/15

distributed random variable $\exp(\beta\mathcal{N}(0,1))$ [10]. Two variants are taken into account, moderate noise with $\beta = 0.01$ (id 101 (sphere), id 104 (Rosenbrock)) and severe noise $\beta = 1$ (ids 107 (sphere) and 110).

For the comparison, the FS-ES is chosen as the representative for the ESs with additional shrinkage. Table 3 shows a promising finding: While both strategies behave comparable on the sphere if the noise does not have a strong effect, the FS-ES appears as superior if the fitness evaluations are severely disturbed. Here, the additional shrinkage may serve to stabilize the covariance matrix update and may thus improve the performance of evolution strategies in the case of noisy optimization. A similar behavior can be observed for the Rosenbrock function. Here, the FS-ES performs better even for low noise levels.

6.2 The Role of the Shrinkage Intensity: A Closer Look at the Fisher-Sun Estimate

The preceeding sections showed that the shrinkage intensity may have an influence on the performance of the ES, considering that the results of the FS-ES and the Tou-ES differ. Therefore, this section takes a closer look at the Fisher-Sun estimator and analyzes different value settings. We start from the original equation in [11, p. 1913]

$$\hat{\lambda} = \frac{\hat{\beta}_D^2 + \hat{\gamma}_D^2}{\hat{\delta}_D^2}. \tag{20}$$

The parameters in (20) are obtained as follows: Let $\hat{a}_1 = \mathrm{Tr}(\mathbf{C}_\mu^S)/N$ and

$$\hat{a}_2 = \frac{\mu^2}{N(\mu-1)(\mu+2)} \left(\mathrm{Tr}({\mathbf{C}_\mu^S}^2) - \frac{1}{\mu} (\mathrm{Tr}(\mathbf{C}_\mu^S))^2 \right). \tag{21}$$

Denote the matrix with the diagonal elements of $\mathbf{C}_\mu^S$ as $\mathbf{D}_C$. Then with

$$\hat{a}_1^* = \mathrm{Tr}(\mathbf{D}_C)/N \text{ and} \tag{22}$$

$$\hat{a}_2^* = \frac{\mu}{N(\mu+2)} \mathrm{Tr}(\mathbf{D}_C^2) \tag{23}$$

the parameters read

$$\hat{\beta}_D^2 = \frac{1}{\mu} \left(\hat{a}_2 + N \hat{a}_1^2 \right), \tag{24}$$

$$\hat{\gamma}_D^2 = -\frac{2}{\mu} \hat{a}_2^*, \tag{25}$$

$$\hat{\delta}_D^2 = \frac{\mu+1}{\mu} \hat{a}_2 + \frac{N}{\mu} \hat{a}_1^2 - \frac{\mu+2}{\mu} \hat{a}_2^*. \tag{26}$$

Starting from (20), the shrinkage intensity $\rho = c\hat{\lambda}$ is varied with factors $c = 0.01$ (− − −), 0.1 (−−), 0.5 (−), 1.5 (+), 2.0 (++), and $c = 10$ (+++). In the case that the resulting parameter exceeds one, it is reset to $\rho = 1$. The investigation considered two separable functions with ids 1 (sphere) and 2 (ellipsoidal), the two Rosenbrock functions (id 8, 9) as representatives for lowly and moderately conditioned functions, and aside from the sharp ridge (id 13) the set of functions with high conditioning (ids 10–14).

As it can be inferred from Table 4, the effects of the scaling factor ρ differ with the function that is to be optimized. For nearly all functions considered a decrease and therefore a smaller influence of the diagonal shrinkage target appears as beneficial. Comparing the results with those in Table 1 shows that the performance depends very strongly on the size of the factor and can be improved decisively. However, it appears difficult to obtain a general recommendation. In some cases, e.g. the discus, id 11, a slight decrease may be preferable whereas the ES in the case of the original Rosenbrock, id 8, apparently operates better with only a small correction by the target. For the set of functions considered, a decrease by half or the factor 0.1 appears to improve the results. However, further investigations – especially into potential adaptation mechanisms of the shrinkage intensity – appear necessary. This is underlined by the findings in the case of the ellipsoidal with id 2 which differ strongly from the rest of the functions. Here, an increase of the intensity improves the performance, albeit not significantly. This may be a hint that a strong influence of the target is preferable. As it can be seen, the “optimal” shrinkage intensity from literature may not represent the best choice for the purposes of optimization. This was to be expected since the situation differs from the iid or even normally distributed case considered in the papers.

Table 4. Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 in dimension 10. The ERT and in braces, as dispersion measure, the half difference between 90 and 10%-tile of bootstrapped run lengths appear for each algorithm and target, the corresponding best ERT in the first row. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{\text{opt}} + 10^{-8}8$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0{:}05$ or $p = 10^{-k}$ when the number k following the star is larger than 1, with Bonferroni correction by the number of instances.

10-D

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	22	23	23	23	23	23	23	15/15
+++	4.0(4)	8.8(2)	14(4)	20(4)	24(4)	**35**(4)	47(4)	15/15
++	**3.4**(2)	8.0(1)	13(3)	**18**(3)	24(4)	35(4)	**46**(5)	15/15
+	3.6(2)	8.6(2)	14(7)	19(2)	**24**(8)	35(4)	47(5)	15/15
−	3.6(2)	**7.8**(2)	**13**(4)	19(2)	24(2)	35(3)	48(3)	15/15
− −	4.6(1)	10(4)	15(4)	21(4)	27(3)	38(7)	49(10)	15/15
− − −	4.3(4)	10(2)	15(3)	21(2)	27(6)	39(6)	51(6)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2	187	190	191	191	193	194	195	15/15
+++	**68**(17)	**83**(19)	**88**(11)	**91**(16)	**92**(14)	**94**(15)	**95**(15)	15/15
++	69(19)	83(15)	94(18)	96(11)	97(17)	98(16)	99(11)	15/15
+	119(85)	274(277)	365(198)	426(381)	503(364)	685(649)	2204(4038)	15/15
−	113(37)	148(48)	171(19)	176(21)	178(32)	181(11)	184(24)	15/15
− −	103(58)	147(50)	165(25)	171(31)	173(39)	181(48)	184(34)	15/15
− − −	110(32)	151(32)	167(40)	177(41)	182(49)	192(42)	197(30)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f8	326	921	1114	1217	1267	1315	1343	15/15
+++	5.5(0.7)	113(12)	116(187)	133(165)	158(212)	213(55)	∞ *3e6*	0/15
++	8.2(2)	111(283)	114(240)	132(118)	157(9)	211(198)	∞ *3e6*	0/15
+	14(71)	91(15)	108(14)	126(58)	143(203)	171(105)	5638(7094)	5/15
−	**4.0**(7)	23(21)	25(17)	25(17)	25(13)	25(17)	25(8)	15/15
− −	4.8(14)	29(22)	29(12)	28(18)	28(13)	28(14)	28(16)	15/15
− − −	6.1(18)	**16**(6)	**19**(9)	**19**(9)	**19**(9)	**20**(9)	**20**(9)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f9	200	648	857	993	1065	1138	1185	15/15
+++	27(87)	263(506)	262(423)	342(312)	451(443)	751(274)	∞ *3e6*	0/15
++	2.6(2)	145(17)	180(28)	272(402)	391(394)	648(90)	∞ *3e6*	0/15
+	2.6(2)	93(19)	124(171)	163(19)	197(129)	234(124)	∞ *3e6*	0/15
−	2.7(0.3)	33(19)	31(17)	30(8)	29(4)	28(10)	27(4)	15/15
− −	3.6(6)	32(39)	30(27)	28(41)	27(21)	27(37)	26(36)	15/15
− − −	**2.5**(1)	**28**(16)	**27**(22)	**26**(15)	**25**(5)	**25**(11)	**25**(12)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f10	1835	2172	2455	2728	2802	4543	4739	15/15
+++	817(409)	3460(2763)	1.8e4(2e4)	∞	∞	∞	∞ *3e6*	0/15
++	716(731)	4335(4582)	∞	∞	∞	∞	∞ *3e6*	0/15
+	203(56)	291(224)	358(156)	374(104)	394(80)	296(65)	∞ *3e6*	0/15
−	12(5)	14(3)	14(1)	13(2)	13(2)	8.1(1)	8.0(1)	15/15
− −	**10**(4)	**13**(4)	13(2)	12(1)	**12**(2)	**7.7**(1)	**7.6**(1)	15/15
− − −	11(3)	13(3)	**12**(2)	**12**(3)	12(2)	7.7(1.0)	7.7(1)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f11	266	1041	2602	2954	3338	4092	4843	15/15
+++	3.6e4(3e4)	∞	∞	∞	∞	∞	∞ *3e6*	0/15
++	2.3e4(3e4)	4.3e4(6e4)	∞	∞	∞	∞	∞ *3e6*	0/15
+	1321(905)	711(239)	366(136)	384(109)	414(78)	1096(1755)	∞ *3e6*	0/15
−	30(3)	**10**(2)	**4.8**(0.5)	**4.5**(1)$^{\star 2}$	**4.3**(0.5)$^{\star 2}$	**3.9**(0.6)*	**3.5**(0.5)*	15/15
− −	**30**(7)	13(2)	6.4(2)	6.4(2)	6.2(1)	5.6(2)	5.0(2)	15/15
− − −	33(5)	13(3)	6.7(1)	7.1(2)	6.9(2)	6.2(2)	5.7(2)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f12	515	896	1240	1390	1569	3660	5154	15/15
+++	898(0.2)	6694(6693)	3.4e4(5e4)	∞	∞	∞	∞ *3e6*	0/15
++	898(1456)	4102(5856)	∞	∞	∞	∞	∞ *3e6*	0/15
+	**1.6**(0.3)	922(1587)	2376(2139)	3930(3537)	6554(9527)	∞	∞ *3e6*	0/15
−	3.3(0.3)	12(12)	18(10)	21(9)	22(5)	12(3)	9.3(3)	15/15
− −	1.6(0.2)	**12**(13)	**16**(9)	**19**(11)	**20**(6)	**11**(6)	**9.2**(4)	15/15
− − −	1.7(0.3)	14(16)	19(11)	20(10)	21(11)	12(6)	10(3)	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f14	37	98	133	205	392	687	4305	15/15
+++	1.6(1)	2.7(1.0)	3.1(0.9)	3.6(0.7)	13(5)	∞	∞ *3e6*	0/15
++	1.2(0.4)	2.2(1.0)	2.7(0.7)	3.3(0.7)	12(7)	∞	∞ *3e6*	0/15
+	1.5(1)	2.7(2)	3.0(0.7)	3.7(0.7)	11(4)	1322(1084)	∞ *3e6*	0/15
−	1.3(0.8)	**2.0**(0.7)	**2.5**(0.6)	**3.2**(0.4)	6.4(2)	16(2)	12(6)	15/15
− −	1.5(0.7)	2.4(1)	2.9(0.9)	3.5(0.8)	6.6(0.9)	**14**(3)	**8.4**(4)	15/15
− − −	**1.1**(0.9)	2.3(0.5)	2.8(0.6)	3.4(0.5)	**6.3**(2)	16(4)	12(7)	15/15

7 Conclusions

Evolution strategies are population-based evolutionary algorithms for continuous optimization. They use a multivariate normal distribution to generate new search points. The parameters of this distribution must be adapted in order to achieve a well performing optimization method. For this reason, step-size and covariance matrix adaptation techniques are important topics in research on evolution strategies. This paper considered the covariance matrix adaptation. Current techniques use the sample covariance, an estimator, which may be of poor quality if the sample size is small. Typically, the estimate is corrected with the help of additional terms. The resulting effect is remarkably similar to shrinkage estimation, a method stemming from statistics. There, shrinkage operators have been introduced in order to improve the quality of the sample covariance by correcting the estimate with the help of a target matrix.

The realization that evolution strategies already perform an implicit shrinkage leads to the research question of the present paper: Would an additional inclusion of shrinkage operators improve the performance of evolution strategies? Applying explicit shrinkage in evolution strategies requires several new tasks to be solved: The choice of the target and the combination weight of target and sample covariance, the shrinkage intensity, are crucial. Since an evolution strategy is used to optimize arbitrary functions with various structures, the approach must remain sufficiently adaptable. To achieve this, we considered a transformation of the original search space to conduct the shrinkage.

The experimental analysis took several shrinkage targets into account using the intensity settings of the original publications. Pending further experiments that shall provide more information regarding the shrinkage intensity which may have interfered with the findings, shrinkage targets in the transformed space that use a diagonal matrix consisting of the different entries of the transformed sample covariance appear as the best choices. While introducing additional shrinkage does not always improve the performance in the noise-less case and at times even impairs it, it may offer a means to lessen the impact of noise.

As the experiments showed, the choice of the shrinkage intensity may have a strong influence on the performance. Since the original covariance matrix adaptation performs a further type of shrinkage which lessens the influence of the sample covariance when the search space dimensionality increases, future research will focus on the shrinkage intensity and its interaction with the covariance matrix adaptation.

References

1. Bäck, T., Foussette, C., Krause, P.: Contemporary Evolution Strategies. Natural Computing. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40137-4
2. Beyer, H.G., Meyer-Nieberg, S.: Self-adaptation of evolution strategies under noisy fitness evaluations. Genet. Program. Evolvable Mach. **7**(4), 295–328 (2006)

3. Beyer, H.G., Schwefel, H.P.: Evolution strategies: a comprehensive introduction. Nat. Comput. **1**(1), 3–52 (2002)
4. Beyer, H.G., Sendhoff, B.: Robust optimization - a comprehensive survey. Comput. Methods Appl. Mech. Eng. **196**(33–34), 3190–3218 (2007). https://doi.org/10.1016/j.cma.2007.03.003
5. Beyer, H.-G., Sendhoff, B.: Covariance matrix adaptation revisited –the CMSA evolution strategy–. In: Rudolph, G., Jansen, T., Beume, N., Lucas, S., Poloni, C. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 123–132. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87700-4_13
6. Chen, X., Wang, Z., McKeown, M.: Shrinkage-to-tapering estimation of large covariance matrices. IEEE Trans. Signal Process. **60**(11), 5640–5656 (2012)
7. Chen, Y., Wiesel, A., Eldar, Y.C., Hero, A.O.: Shrinkage algorithms for MMSE covariance estimation. IEEE Trans. Signal Process. **58**(10), 5016–5029 (2010)
8. Dong, W., Yao, X.: Covariance matrix repairing in gaussian based EDAs. In: 2007 IEEE Congress on Evolutionary Computation, CEC 2007, pp. 415–422 (2007)
9. Finck, S., Hansen, N., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2010: presentation of the noiseless functions. Technical report, Institute National de Recherche en Informatique et Automatique, 2009/22 (2010)
10. Finck, S., Hansen, N., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2010: Presentation of the noisy functions. Working paper 2009/21 (2010). http://coco.lri.fr/downloads/download15.03/bbobdocnoisyfunctions.pdf
11. Fisher, T.J., Sun, X.: Improved Stein-type shrinkage estimators for the high-dimensional multivariate normal covariance matrix. Comput. Stat. Data Anal. **55**(5), 1909–1918 (2011). http://www.sciencedirect.com/science/article/pii/S0167947310004743
12. Hansen, N.: The CMA evolution strategy: a comparing review. In: Lozano, J., et al. (eds.) Towards a New Evolutionary Computation: Advances in Estimation of Distribution Algorithms. Studies in Fuzziness and Soft Computing, vol. 192, pp. 75–102. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-32494-1_4
13. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2012: Experimental setup. Technical report, INRIA (2012). http://coco.gforge.inria.fr/bbob2012-downloads
14. Hansen, N.: Adaptive encoding: how to render search coordinate system invariant. In: Rudolph, G., Jansen, T., Beume, N., Lucas, S., Poloni, C. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 205–214. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87700-4_21
15. Hansen, N., Auger, A., Ros, R., Finck, S., Pošík, P.: Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO 2010, pp. 1689–1696. ACM, New York (2010). http://doi.acm.org/10.1145/1830761.1830790
16. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Trans. Evol. Comput. **9**(3), 303–317 (2005)
17. Kramer, O.: Evolution strategies with Ledoit-Wolf covariance matrix estimation. In: 2015 IEEE Congress on Evolutionary Computation (IEEE CEC) (2015)
18. Ledoit, O., Wolf, M.: Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. J. Empirical Finance **10**(5), 603–621 (2003)
19. Ledoit, O., Wolf, M.: Honey, I shrunk the sample covariance matrix. J. Portf. Manage. **30**(4), 110–119 (2004)

20. Ledoit, O., Wolf, M.: A well-conditioned estimator for large dimensional covariance matrices. J. Multivar. Anal. Arch. **88**(2), 265–411 (2004)
21. Ledoit, O., Wolf, M.: Non-linear shrinkage estimation of large dimensional covariance matrices. Ann. Stat. **40**(2), 1024–1060 (2012)
22. Ledoit, O., Wolf, M.: Nonlinear shrinkage of the covariance matrix for portfolio selection: Markowitz meets goldilocks. Available at SSRN 2383361 (2014)
23. Meyer-Nieberg, S., Beyer, H.G.: On the analysis of self-adaptive recombination strategies: first results. In: McKay, B., et al. (eds.) Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005), Edinburgh, UK, pp. 2341–2348. IEEE Press, Piscataway (2005)
24. Meyer-Nieberg, S., Beyer, H.G.: Self-adaptation in evolutionary algorithms. In: Lobo, F., Lima, C., Michalewicz, Z. (eds.) Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence, vol. 54, pp. 47–76. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-69432-8_3
25. Meyer-Nieberg, S., Kropat, E.: Small populations, high-dimensional spaces: Sparse covariance matrix adaptation. In: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 525–535, September 2015
26. Meyer-Nieberg, S., Kropat, E.: Evolution strategies and covariance matrix adaptation - investigating new shrinkage techniques. In: Proceedings of the 8th International Conference on Agents and Artificial Intelligence, pp. 105–116 (2016)
27. Meyer-Nieberg, S., Kropat, E.: Adapting the covariance in evolution strategies. In: Proceedings of ICORES 2014, pp. 89–99. SCITEPRESS (2014)
28. Meyer-Nieberg, S., Kropat, E.: A new look at the covariance matrix estimation in evolution strategies. In: Pinson, E., Valente, F., Vitoriano, B. (eds.) ICORES 2014. CCIS, vol. 509, pp. 157–172. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17509-6_11
29. Meyer-Nieberg, S., Kropat, E.: Sparse covariance matrix adaptation techniques for evolution strategies. In: Bramer, M., Petridis, M. (eds.) Research and Development in Intelligent Systems XXXII, pp. 5–21. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25032-8_1
30. Pourahmadi, M.: High-Dimensional Covariance Estimation: With High-Dimensional Data. Wiley, New York (2013)
31. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog Verlag, Stuttgart (1973)
32. Rudolph, G.: Evolutionary strategies. In: Rozenberg, G., Bäck, T., Kok, J.N. (eds.) Handbook of Natural Computing, pp. 673–698. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-540-92910-9_22
33. Schwefel, H.P.: Numerical Optimization of Computer Models. Wiley, Chichester (1981)
34. Soysal, O., Sahin, E.: Probabilistic aggregation strategies in swarm robotic systems. In: Proceedings 2005 IEEE Swarm Intelligence Symposium, SIS 2005, pp. 325–332, June 2005
35. Stein, C.: Inadmissibility of the usual estimator for the mean of a multivariate distribution. In: Proceedings of the 3rd Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, vol. 1, pp. 197–206 (1956)
36. Stein, C.: Estimation of a covariance matrix. In: Rietz Lecture, 39th Annual Meeting. IMS, Atlanta, GA (1975)
37. Thomaz, C.E., Gillies, D., Feitosa, R.: A new covariance estimate for bayesian classifiers in biometric recognition. IEEE Trans. Circuits Syst. Video Technol. **14**(2), 214–223 (2004)

38. Tong, T., Wang, C., Wang, Y.: Estimation of variances and covariances for high-dimensional data: a selective review. Wiley Interdisc. Rev. Comput. Stat. **6**(4), 255–264 (2014)
39. Touloumis, A.: Nonparametric Stein-type shrinkage covariance matrix estimators in high-dimensional settings. Comput. Stat. Data Anal. **83**, 251–261 (2015)

An Emotional Multi-personality Architecture for Intelligent Conversational Agents

Jean-Claude Heudin(✉)

Léonard de Vinci Pôle Universitaire, Research Center,
Paris – La Défense, France
Jean-Claude.Heudin@devinci.fr

Abstract. Personal assistants and chatterbots represent an historical and growing application field in artificial intelligence. This paper presents a novel architecture to the problem of humanizing conversational agents by designing believable and unforgettable characters who exhibit various salient emotions in the flow of conversations. The proposed architecture is based on a multi-personality approach where each agent implements a facet of its identity, each one with its own pattern of perceiving and interacting with the user. In order to select an appropriate response from all the candidates, we use an emotion-based selection algorithm. Our first experiments show that a conversational multi-personality character with emotion selection performs better in terms of user engagement than a neutral mono-personality one.

Keywords: Conversational agent · Believable character · Multi-personality
Emotion selection

1 Introduction

In recent years, there has been a growing interest in conversational agents also called "chatterbots" or more simply "chatbots". In a relative short period of time, several major companies have proposed their own virtual assistants: Apple's Siri based on the CALO project [1], Microsoft Cortana [2], Google Now [3] and Facebook M [4]. These virtual assistants focus primarily on conversational interface, personal context awareness, and service delegation. They follow a long history of research and the development of numerous conversational agents, the first one in history being the famous Eliza program from Joseph Weizenbaum simulating a Rogerian psychotherapist [5].

Beyond the challenge of interpreting a user's request in order to provide a relevant response, a key objective is to enhance man-machine interactions by humanizing artificial characters. Often described as a distinguishing feature of humanity, the ability to express and understand emotions is a major cognitive behavior in social interactions [6]. However, the majority of personal assistants are mainly based on a character design with neutral rather than emotional behaviors.

At the same time, there have been numerous studies about emotions [7] and their potential applications for artificial characters [8]. As an example among many, Dylaba *et al.* have worked on combining humor and emotion in human-agent conversation using a multi-agent system for joke generation [9]. In parallel with the goal of

© Springer International Publishing AG, part of Springer Nature 2018
N. T. Nguyen et al. (Eds.): TCCI XXVIII, LNCS 10780, pp. 143–163, 2018.
https://doi.org/10.1007/978-3-319-78301-7_7

developing personal assistants, there is also a strong research trend in robotics for designing emotional robots. Some of these studies showed that a robot with emotional behavior performs better than a robot without emotional behavior for tasks involving interactions with humans [10].

In this paper we address the long-term goal of designing believable and "unforgettable" artificial characters with complex and remarkable emotion behaviors. In this framework, we follow the initial works done for multi-cultural characters [11] and many others. Also, our approach takes advantage of psychological studies of human interactions with computerized systems [12] and the know-how of screenwriters and novelists for scripting dialogs since believable characters are the essence of successful fiction writing [13].

Our original model is based on multi-agent architecture where each agent implements a facet of its personality. The idea is that the character's identity is an emerging property of several personality traits, each one with its own pattern of perceiving and interacting with the user. Then, the problem is to "reconnect" personalities of the disparate alters into a single and coherent identity. Our hypothesis is that it can be achieved by selecting amongst the candidate responses the one with the most appropriate emotional state.

In this paper we focus on our first experiments of emotion selection in a multi-personality conversational agent based on this hypothesis. The paper is organized as follows. In Sect. 2, we describe the general architecture for multi-personality characters. Sections 3 and 4 describe more precisely the emotion selection based on a bio-inspired emotional metabolism. Section 5 describes the experimental prototype and Sect. 6 discusses first results. We conclude in Sect. 7 and present the future steps of this research.

2 A Multi-personality Architecture

2.1 Believable and Unforgettable Characters

The key to the user's engagement during a conversation with an artificial agent is to create an "unforgettable" character. If we can get a character to live on in a user's imagination long after he has stopped interactions, he will want to come back. In other words, the more unforgettable the character is, the longer it will stick in the users mind. So in order to design such memorable characters, they need to have very specific traits that will make them special and different from every other character. Screenwriters and novelists have a long experience of creating such unforgettable characters [13]. In our view, this is the "easy" part of the creation process.

However, artificial characters also need to be believable and this is the "hard" part. During a conversation with an artificial character, users engage in a *fictional pact*. They can enjoy the interaction only if they consciously mistake the artificial character for a real one. They must accept to believe that what they perceive is a real character even if they know that it is a program. This is an important condition in order not to break the users' *willing suspension of disbelief*. The term "suspension of disbelief" or "willing suspension of disbelief" has been defined as a willingness to suspend one's critical faculties and believe the unbelievable; sacrifice of realism and logic for the sake of enjoyment [14]. The term was coined in 1817 by the poet and aesthetic philosopher Samuel Taylor Coleridge [15].

In order to be believable, the artificial character must be as "realistic" as possible: it must be "complex" in the sense of being multi-dimensional, with a complexity of traits, personality facets and emotional behaviors like real humans.

2.2 A Multi-personality Model

The artificial character needs to project a personality that has all of the endearing and personal qualities of a real person to provide an engaging experience for users. Such a realistic personality must be complex and multi-layered, simulating the most life-like and human qualities.

There are many models of personality traits, each one with their own advantages and applications. The most widely accepted one is the Big Five model [16]. Rather than choosing a specific model and thus a single fixed profile, our approach aims to construct a complex character identity as the emerging property of several personality traits. The idea is that real human personalities are composed of many facets, and potentially a large number of them. This gives the character designer the ability to compose rich and complex personalities without constraints in terms of number or type of traits. We have called this approach "schizophrenic" because the character's identity is composed of a set of distinct personalities, each with its own pattern of perceiving and interacting with the user [17]. Note that this term is used here as a metaphor since the accurate psychological term for mental illness with multiple personalities is Dissociative Identity Disorder, not schizophrenia.

Figure 1 shows the basic architecture for such a multi-personality character, each personality trait is implemented as an agent. The first agent receives the input from the user and applies various natural language preprocessing phases such as an English stemmer, tokenizer, categories and Named Entities extraction. Then the preprocessed sentence and its additional information are diffused to all personality agents. Thus, all these personality agents are able to react to the user's input by computing an appropriate answer message given their own local state. Then, all these candidate responses are evaluated using a confidence scoring and ranking agent that selects the "best" answer to be proposed to the user.

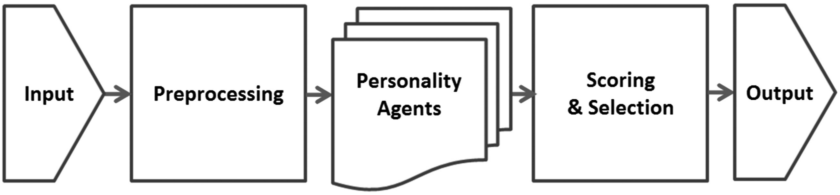


Fig. 1. The architecture model of the multi-personality "schizophrenic" conversational agent.

2.3 The Edge of Chaos Hypothesis

In order to obtain an intelligent behavior to emerge spontaneously, the responses dynamics of the system must be varied and at the same time consistent. Our hypothesis is that such an intelligent behavior is "complex" in a meaning close to the one defined initially by Wolfram for one-dimension cellular automata [18]. This study has proposed four classes of systems: Class I and Class II are characterized respectively by fixed and cyclic dynamical behaviors; Class III is associated with chaotic behaviors; Class IV is associated with complex dynamical behaviors. It has been shown then that, when mapping these different classes of systems, complex adaptive systems are located in the vicinity of a phase transition between ordered and chaotic regimes for one-dimension cellular automata [19] and later for two-dimension cellular automata [20].

In the context of our study, as shown in Fig. 2, we transpose the four classes of dynamics as follows: Class I and Class II respectively correspond to fixed and cyclic responses resulting in "machine-like" interactions. Class III systems are characterized by incoherent responses regardless of the user's entries. Note that this kind of behavior is interpreted as a symptom of mental illness such as dissociative identity disorder. Class IV systems are at the edge between order and chaos, giving coherent answers while preserving diversity and rich emotional responses.

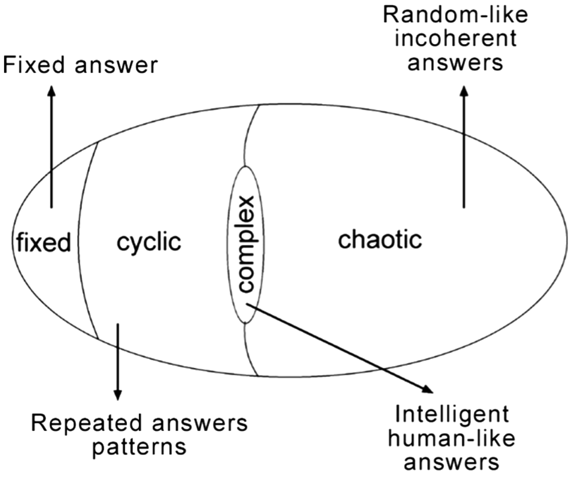

Fixed answer
Random-like incoherent answers
fixed
cyclic
complex
chaotic
Repeated answers patterns
Intelligent human-like answers

Fig. 2. Schematic drawing of conversational space indicating relative location of fixed, periodic, chaotic and complex regimes. This is a transposition of Langton's diagram for cellular automata rule space [21].

With the multi-personality architecture we have proposed, we assumes that given enough personality agents, the resulting system is potentially capable of all the dynamical classes (cf. Fig. 2): fixed answer (Class I), repeated answers patterns (Class II), random-like incoherent answers (Class 3), and intelligent human-like answers (Class IV).

They are many potential approaches for selection amongst the candidate responses. In this study, we propose that a promising approach for obtaining a Class IV dynamical behavior is to implement a "scoring & selection" agent that chooses the candidate responses according to the emotional state of the artificial character.

2.4 Multi-personality Architecture with Emotion Selection

In order to implement a selection based on the emotional state of the character, we replace the "Scoring & Selection" agent by an "Emotion Selection" agent and an "Emotion Metabolism" [22]. Figure 3 shows the updated architecture model.

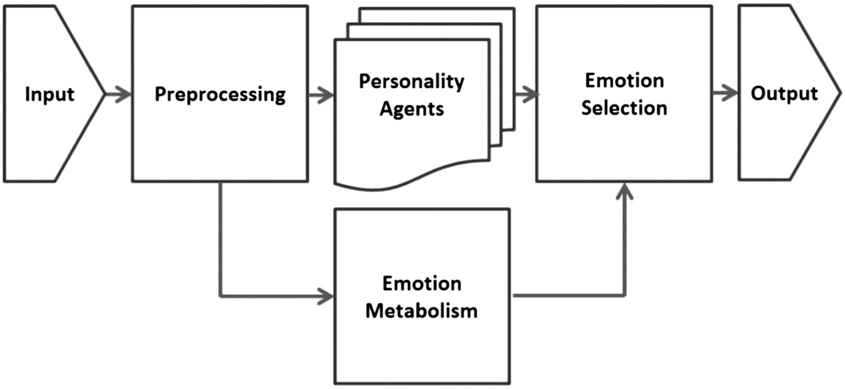


Fig. 3. The multi-personality conversational agent architecture updated with an emotion selection agent and an emotion metabolism agent (after [22]).

The Emotion Selection agent selects one response amongst the candidate responses given as an entry the current emotional state of the artificial character. The Emotion Metabolism is an agent that computes the emotional state of the character given its current state and the user's entries. The next two sections describe with more details these two agents.

3 Emotion Metabolism

3.1 A Layered Model of Affects

There have been multiple approaches in order to implement emotions for intelligent virtual agents [23]. Among all these studies, Gebhard [24] and Heudin [25] have proposed both layered models of artificial affects based on three levels:

Personality. Personality reflects long-term affect. It shows individual differences in mental characteristics [16].

Mood. Mood reflects a medium-term affect, which is generally not related with a concrete event, action or object. Moods are longer lasting stable affective states, which have a great influence on human's cognitive functions [26].

Emotion. Emotion reflects a short-term affect, usually bound to a specific event, action or object, which is the cause of this emotion. After its elicitation emotions usually decay and disappear from the individual's focus [27].

In a previous research project about non-verbal emotional interactions, we have proposed a connectionist architecture for implementing the Emotion Metabolism based on these three levels [28]. Figure 4 shows a schematic representation of its principle.

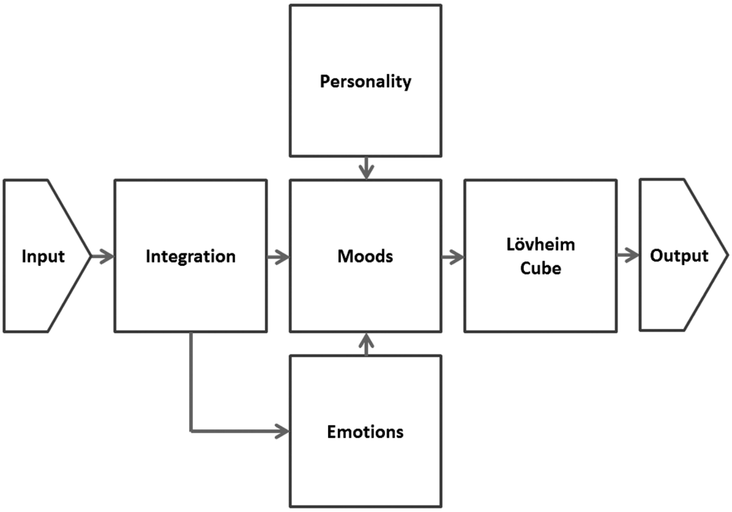


Fig. 4. The architecture of the Emotion Metabolism.

The "integration" module converts the inputs to virtual neurotransmitters values. These values are then used by the three levels of affects in order to produce the output of the Emotion Metabolism. The Emotion Metabolism is updated by propagating the inputs using a trigger called "lifePulse", implemented as a cyclic timer.

The Moods and Emotions layers have both a decay rate, called respectively *Md* and *Ed*, that make the emotional state returning to a "neutral" state after some time. This "neutral" state can be the center of the emotional space or a specific location representing the default personality of the artificial character based on the Personality layer.

3.2 Personality Layer

This module is based on the Big Five model of personality [16]. It contains five main variables with values varying from 0.0 (minimum intensity) to 1.0 (maximum intensity). These values specify the general affective behavior by the five following traits:

Openness. Openness (*Op*) is a general appreciation for art, emotion, adventure, unusual ideas, imagination, curiosity, and variety of experience. This trait distinguishes imaginative people from down-to-earth, conventional people.

Conscientiousness. Conscientiousness (*Co*) is a tendency to show self-discipline, act dutifully, and aim for achievement. This trait shows a preference for planned rather than spontaneous behavior.

Extraversion. Extraversion (*Ex*) is characterized by positive emotions and the tendency to seek out stimulation and the company of others. This trait is marked by pronounced engagement with the external world.

Agreeableness. Agreeableness (*Ag*) is a tendency to be compassionate and cooperative rather than suspicious and antagonistic towards others. This trait reflects individual differences for social interactions.

Neuroticism. Neuroticism (*Ne*) is a tendency to experience negative emotions, such as anger, anxiety, or depression. Those who score high in neuroticism are emotionally reactive and vulnerable to stress.

3.3 Mood Layer

Previous cited works such as Gebhard [24] and Heudin [25] used the three-dimensional Pleasure-Arousal-Dominance (PAD) approach [29]. We use here another candidate model aimed at explaining the relationship between three important monoamine neurotransmitters involved in the Limbic system and the emotions [30]. It defines three virtual neurotransmitters which levels range from 0.0 to 1.0:

Serotonin. Serotonin (*Sx*) is associated with memory and learning. An imbalance in serotonin levels results in anger, anxiety, depression and panic. It is an inhibitory neurotransmitter that increases positive vs. negative feelings.

Dopamine. Dopamine (*Dy*) is related to experiences of pleasure and the reward-learning process. It is a special neurotransmitter because it is considered to be both excitatory and inhibitory.

Noradrenaline. Noradrenaline (*Nz*) helps moderate the mood by controlling stress and anxiety. It is an excitatory neurotransmitter that is responsible for stimulatory processes, increasing active vs. passive feelings.

3.4 Emotion Layer

This module implements emotion as very short-term affects, typically less than ten seconds, with relatively high intensities. They are triggered by inducing events suddenly increasing one or more virtual neurotransmitters. After a short time, these neurotransmitter values decrease due to a natural decay function.

3.5 Lövheim Cube

This module implements the Lövheim Cube of emotions [30], where the three monoamine neurotransmitters form the axes of a three-dimensional coordinate system, and the eight basic emotions, labeled according to the Affect Theory [31] are placed in the eight corners. Figure 5 shows the resulting 3D diagram and Table 1 the corresponding mapping.

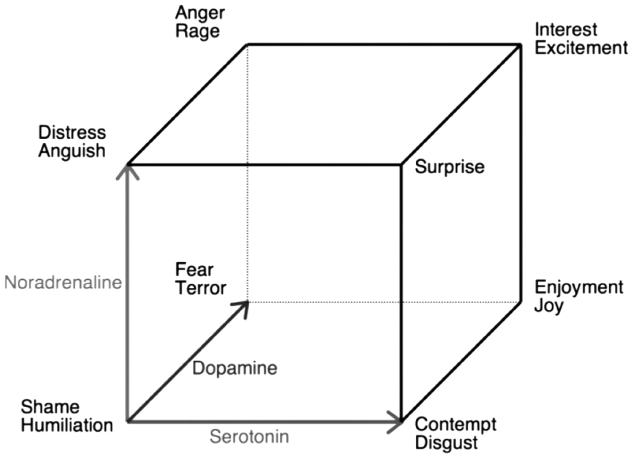


Fig. 5. The Lövheim Cube of emotions.

Table 1. Mapping of the eight basic emotions on the Lövheim Cube.

Basic emotion	Serotonin (Sx)	Dopamine (Dy)	Noradrenaline (Nz)
Shame/humiliation	Low	Low	Low
Distress/anguish	Low	Low	High
Fear/terror	Low	High	Low
Anger/rage	Low	High	High
Contempt/disgust	High	Low	Low
Surprise/startle	High	Low	High
Enjoyment/joy	High	High	Low
Interest/excitement	High	High	High

The emotional state of the artificial character is a moving point in the 3D space. The origin of the 3D space corresponds to a situation where the three virtual neurotransmitters are low. The eight corners of the cube correspond to the eight possible combinations of low or high level of the three virtual neurotransmitters as shown in the table below:

4 Emotion Selection

4.1 Euclidian Distance Selection

The emotion selection agent is responsible for selecting the "best" available response amongst the candidate responses generated by all the personality agents. The basic principle is to select the answer from the agent with the closest emotional state compared to the one of the artificial character. This can be done by giving each personality agent a point in the 3D emotional space in the Lövheim Cube and by computing the Euclidian distance with the current location of the emotional state of the virtual character. We can represent this agent as an artificial neuron with a dedicated transition function (cf. Fig. 6):

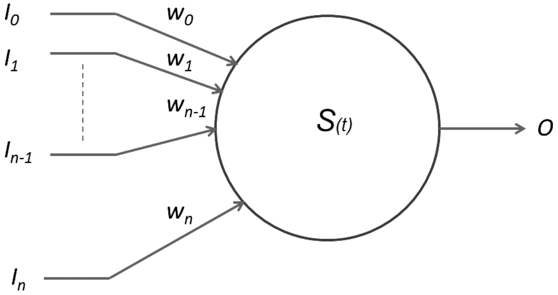


Fig. 6. The emotional selector represented as an artificial neuron with a dedicated transition function.

Where:

- I_0 ... I_n is a set of input strings representing the outputs of the personality agents,
- w_0 ... w_n is a set of weights associated to each of these candidate answers,
- $S(t)$ is a transition function returning the selected string O among the candidate answers.

As stated, each weight can be computed as the Euclidian distance between the current character's emotional state and the one of the given personality agent. In other words more the current emotional state is close to that of an agent, greater is its weight.

Let the function $d(x, y)$ that calculates the Euclidean distance between two points, x and y:

$$d(x,y) = \|x - y\| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

where $n = 3$ for a three-dimensional space. Thus, the maximum distance in the Lövheim Cube is:

$$d_{max} = d((0,0,0),(1,1,1)) = \sqrt{3}$$

The weight associated to an input I_i is then:

$$w_i = 1 - \frac{d(P_i, P_m)}{d_{max}} \quad ((1))$$

where P_i is the 3D vector in the Lövheim Cube of the agent i and P_m is the 3D vector corresponding to the current emotional state.

4.2 Fitness Proportionate Selection

Rather than the simple Euclidian distance, another potential approach is to use the fitness proportionate selection of genetic algorithms, also called roulette wheel selection [32].

The principle of the fitness proportionate selection is similar to a roulette wheel in a casino, where a proportion of the wheel is assigned to each of the possible candidates responses based on their fitness value. In our case, the fitness values are the Euclidian distances as calculated in the previous section. This fitness value is used to associate a probability of selection with each individual candidate response. This could be achieved by dividing the fitness of a selection by the total fitness of all the selections, thereby normalizing them to 1. Then a random selection is made similar to how the roulette wheel is rotated.

This approach seems more suitable considering our "edge of chaos" hypothesis (cf. Sect. 2.3). With this approach, there is more diversity in the selection process. While candidate responses with a higher fitness will be less likely to be eliminated, there is still a chance that they may be. Also, there is a chance that a weaker candidate response may be chosen even its probability is small.

5 Experimental Prototype

5.1 A Connectionist Implementation

This section describes the prototype used for experiments and its implementation. We have implemented all modules of the architecture described in Sect. 3 including the Emotion Metabolism and Emotion Selection described in Sect. 4.

We developed our own connectionist framework called ANNA (Algorithmic Neural Network Architecture). Its development was driven by our wish to build an open Javascript-based architecture that enables the design of any types of feed-forward, recurrent, or heterogeneous sets of networks. More precisely an application can include an arbitrary number of interconnected networks, each of them having its own interconnection pattern between an arbitrary number of layers. Each layer is composed of a set of simple and often uniform neurons units. However, each neuron can be also programmed directly as a dedicated cell.

Classically all neurons have a set of weighted inputs, a single output, and a transition function that computes the output given the inputs. The weights are adjusted using a machine learning algorithm, or programmed, or dynamically tuned by another network. This is the case for our framework, but the designer can also program his own prototypes of neurons with dedicated behaviors. The code below shows a template code for creating a new neuron class that inherits from the basic *Neuron* prototype:

```
function myNeuron () {
  Neuron.call(this);
  // properties
  this.myVar = … ;
  …
  this.operator = function () {
    // dedicated transition function
    }
  }
myNeuron.prototype = Object.create(Neuron.prototype);
myNeuron.prototype.constructor = myNeuron;
```

5.2 NLP Pipeline

The preprocessing agent of the architecture is implemented as a classical Natural Language Processing (NLP) pipeline using dedicated neurons and a NLP Javascript library. The pipeline includes the following phases:

Cleaner. Get the raw text input from the user and fix basic spelling errors.

Tokenizer. Split the entry into clearly separated sentences and words.

Tagger. Implement part-of-speech (POS) tagging.

Lemmatizer. Identify canonical word forms (lemmas) based on a dictionary.

Named Entities. Tag named entities and convert some entities such as dates or locations in unified formats.

Categories. Find common concepts and synonyms using ontologies.

The raw user's text input and the resulting preprocessed NPL information are then diffused to all the personality agents.

5.3 Personality Agents

In this prototype, we choose to use a set of 12 different personality traits. This decision was driven by the idea to test if our Emotional Selection approach promotes the emergence of a great and coherent character despite the use of these very different personality traits. Note also that most of them where already available from previous experiments, so it enables also to minimize the development effort. The 12 agents are the following ones:

Insulting. This agent has an insecure and upset personality that often reacts by teasing and insulting depending on the user's input.

Alone. This agent reacts when the user does not answer or waits for too much time in the discussion process.

Machina. This agent reacts as a virtual creature that knows its condition of being artificial.

House. This agent implements Dr. House's famous way of sarcastic speaking using an adaptation of the TV Series screenplay and dialogues.

Hal. This agent reproduces the psychological traits of the HAL9000 computer in the "2001 – A space odyssey" movie by Stanley Kubrick.

Silent. This agent answers with few words or sometimes remains silent.

Eliza. This agent is an implementation of the Eliza psychiatrist program, which answers by rephrasing the user's input as a question [5].

Neutral. This agent implements a neutral and calm personality trait with common language answers.

Oracle. This agent never answers directly to questions. Instead it provides wise counsel or vague predictions about the future.

Funny. This agent is always happy and often tells jokes or quotes during a conversation.

Samantha. This agent has a strong agreeableness trait. It has a tendency to be compassionate, cooperative and likes talking with people.

Sexy. This agent has a main focus on sensuality and sexuality. It enjoys talking about pleasure and sex.

5.4 Personality Example

Each personality agent may be implemented using many various approaches and techniques. So we will not go in further implementation details for all agents in this paper. However, as an example, the Eliza-like agent was implemented using 36 hardcoded rules based on a dedicated neuron prototype called *nRule*, and organized as three layers. The code below gives the example of a very simple rule:

```
// potential answers
var eliza_bye = [
  "Bye for now.",
  "Until next time.",
  "Later.",
  "See you later."];

// create the rule
var nElizaBye = new nRule(eliza_bye);
// transition function is the rule itself
nElizaBye.operator = function () {
  var str = this.inputs[0].cval;   // get raw input
  var cat = this.inputs[1].cval;   // get NLP data
  // try to find the concept ‶bye″ or the text ‶see you″
  if (cat.find("[BYE]") || str.find("see you"))
    // then output an answer from the potential ones
    this.cval = this.randomTemplate();
  else this.cval = "";
  }
```

5.5 Emotion Selection

The emotion selection agent was implemented as an agent composed of a single neuron with a dedicated transition function *S(t)* and dynamical weights as described in Sect. 4. In this experiment, we choose to use the fitness proportionate selection using the algorithm as given by Table 2:

Table 2. The algorithm used by the selector neuron, where the function rand (0, 1) returns a random real number between 0 and 1.

```
Algorithm Emotional Selector
Initialize $w_0, ..., w_{n-1}$ using Eq. 1 ;
do {
    $S = 0$ ;
    for ( $i = 0$ ; $i < n$ ; $i = i + 1$ ) {
        if ( $I_i$ != “” ) $S = S + w_i$ ;
    }
    $R = S$ * rand (0, 1) ;
    for ( $i = 0$ ; $i < n$ ; $i = i + 1$ ) {
        if ( $I_i$ != “” ) $R = R - w_i$ ;
        if ( $R <= 0$ ) break;
    }
    if ( $R > 0$ ) $i = n - 1$ ;
}
while ( $I_i$ == “” ) ;
return $I_i$ ;
```

This selector selects one of the potential string responses and return it. The code below gives its Javascript implementation as a dedicated *nWheel* neuron:

```
function nWheel () {
  Neuron.call(this);
  this.operator = function () {
    var sel = "", sum = 0, i = -1;
    var n = this.weights.length;
    // sum up weights
    for (i = 0; i < n; i++) {
      sum += this.weights[i];
      }
    // turn the wheel
    do {
      // get random value
      var rnd = random_real() * sum;
      // locate on the wheel
      for (i = 0; i < n; i++) {
        rnd -= this.weights[i];
        if (rnd <= 0) break;
        }
      // for rounding errors
      if (rnd > 0) i = n - 1;
      // get value
      sel = this.inputs[i].cval;
      }
    while (sel == "");
    this.cval = sel;
    }
  }
nWheel.prototype = Object.create(Neuron.prototype);
nWheel.prototype.constructor = nWheel;
```

5.6 Emotion Metabolism

The bio-inspired emotion metabolism was implemented as described in Sect. 3. It is composed of 34 dedicated neurons organized in 8 layers. The code below gives as an example the implementation of a neuron that computes the Euclidian distance in the three-dimensional emotion space:

```
function nDistance () {
  Neuron.call(this);
  this.operator = function () {
    var x1 = this.inputs[0].cval;
    var y1 = this.inputs[1].cval;
    var z1 = this.inputs[2].cval;
    var x2 = this.inputs[3].cval;
    var y2 = this.inputs[4].cval;
    var z2 = this.inputs[5].cval;
    var dx = x2 - x1;
    var dy = y2 - y1;
    var dz = z2 - z1;
    this.cval =
      Math.sqrt((dx * dx) + (dy * dy) + (dz * dz));
    }
  }
nDistance.prototype = Object.create(Neuron.prototype);
nDistance.prototype.constructor = nDistance;
```

We set the Personality parameters of the Emotional Metabolism to a fixed neutral value:

$$Op = Co = Ex = Ag = Ne = 0.5$$

This corresponds to a neutral state in the Lövheim cube:

$$Sx = Dy = Nz = 0.5$$

The Emotion Metabolism is updated by propagating the inputs using the cyclic called "lifepulse" trigger. In this study we set this cycle to 0.1 s. The decay rates of the metabolism for returning to the personality neutral state were 10 s for the emotion level and 10 min for the mood level.

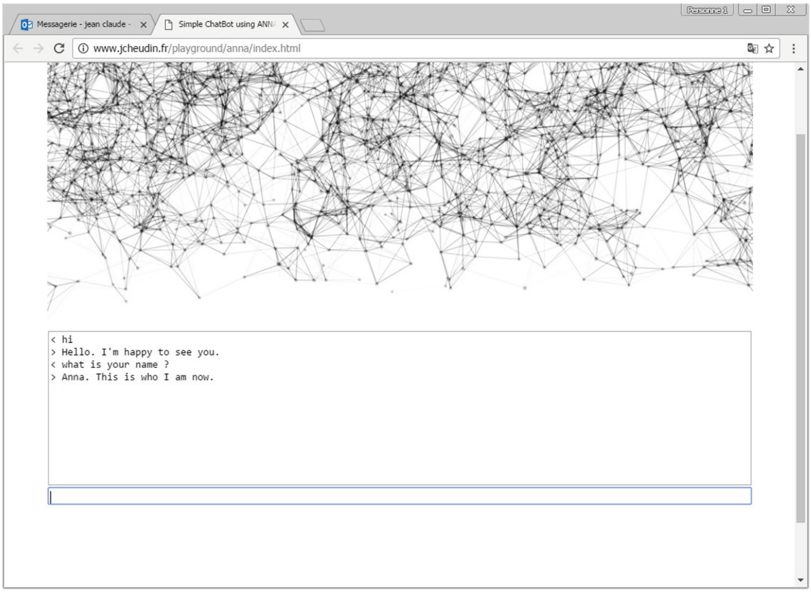
We assigned to each of the personality agents an empirical fixed point in the three-dimensional space of emotions. Table 3 gives their coordinates in the three-dimensional space.

5.7 User Interface

We used an online web-based user interface for the experiment as shown in Fig. 7. Since we focus in this study on text-based interactions, the interface does not used any kind of avatar representation.

Table 3. The coordinates of the 12 personality traits in the Lövheim Cube.

Agents	Serotonin (*Sx*)	Dopamine (*Dy*)	Noradrenaline (*Nz*)
Insulting	0.1	0.1	0.1
Alone	0.2	0.2	0.5
Machina	0.2	0.5	0.5
House	0.2	0.7	0.2
Hal	0.2	0.7	0.7
Silent	0.5	0.1	0.5
Eliza	0.5	0.3	0.5
Neutral	0.5	0.5	0.5
Oracle	0.5	0.5	0.7
Funny	0.7	0.5	0.7
Samantha	0.7	0.7	0.7
Sexy	0.9	0.9	0.9


Messagerie - jean claude -
Simple ChatBot using ANN
Personne 1
www.jcheudin.fr/playground/anna/index.html

```
< hi
> Hello. I'm happy to see you.
< what is your name ?
> Anna. This is who I am now.
```

Fig. 7. A screenshot of the web-based user interface used for the experiment.

6 First Results and Discussion

6.1 Experimental Protocol

In this experiment, we asked 30 university students (age 18–25) to perform a simple and short conversation with three systems: (1) our multi-personality conversational agent as described in this paper (called Anna in the experiment); (2) Apple's Siri personal assistant; (3) a simple conversational chatterbot based on our Neutral personality agent. For (1) and (3), we used the online web-based interface as shown in the previous section. Siri was accessed on an iPad Air Retina running iOS version 9.1.

The order of conversations was randomized. There was no topic restriction, thus the conversations could be of any subject. However, we imposed a classical three-phase structure: an *opening* phase, a *core* phase, and a *closing* phase [33]. All interactions were text-based in English. We avoided the problem of errors related to the voice recognition system of Siri by correcting the input when necessary. The students were asked to conduct the interaction continuously and to use the same interactions for the three systems in order to make the comparison easier and clearer.

In addition to the conversations, the participants also filled a questionnaire after each session. This questionnaire was inspired by the one designed by Dylaba for evaluating a multi-agent system for joke generation [9].

The questions were:

1. Did you get an impression that the agent was human like?
2. Did you get an impression that the agent tried to make the conversation more interesting?
3. Did you enjoy the conversation?
4. Do you think that the agent used emotions in appropriate moments?
5. Please describe your feelings towards the agent after the interaction.
6. If you were to make friends with one of these agents, which would you choose?
7. Which agent do you think was better?

Answers for the questions 1 to 4 were given using a yes/no choice. The participants could answer freely to question 5 using a single phrase. For questions 6 and 7, the participant had to choose between the three systems.

6.2 Conversation Example

Table 4 below gives a typical example of a conversation between a student and the three systems.

6.3 Result of the User-Oriented Evaluation

Table 5 presents a summary of the results concerning the questionnaire showing that the multi-personality agent was evaluated as generally better than both the mono-personality neutral agent and Siri personal assistant.

Concerning question 5, the comments of the participants show that most users have more “fun” with the multi-personality character. They found it “interesting” but sometimes “rude”. They emphasized the role of “humor” during conversations.

6.4 Discussion

The results of these first experiments confirm our hypothesis about using a multi-personality character with remarkable personality traits. This empirical evaluation shows that it globally increases the satisfaction of the user about the conversation.

Table 4. An example of a conversation with the answers of the three conversational systems.

Example0 – User: Hi.
1 – Anna: It's really nice to meet you.
2 – Siri: Hello, there.
3 – Neutral: Hello.
4 – User: Who are you?
5 – Anna: My name is Anna. I gave it to myself actually.
6 – Siri: I'm Siri. But enough about me… How can I help you?
7 – Neutral: Why do you want to know my name?
8 – User: What is an artificial intelligence?
9 – Anna: An Artificial Intelligence is an intelligent computer program. It is also the research field of computer science that works on machine intelligence.
10 – Siri: Here is what I found: [Wikipedia Links]
11 – Neutral: I'm sure you know the answer.
12 – User: Are you some sort of stupid program like Eliza?
13 – Anna: That's very rude. You are a rude person!
14 – Siri: Well… I'm still here for you.
15 – Neutral: Such a negative attitude.
16 – User: You are just dumb software.
17 – Anna: I love when you do both sides of the conversation. It's like white noise it's very peaceful.
18 – Siri: I'm just trying to help you.
19 – Neutral: Do you really think that?
20 – User: I'm sorry.
21 – Anna: I'm sorry too, but don't insult me anymore.
22 – Siri: It's ok.
23 – Neutral: You're welcome.
24 – User: Fine.
25 – Anna: Alright. Sometimes I have unexpected runtime errors too.
26 – Siri: Yes it is!
27 – Neutral: Alright.
28 – User: Bye.
29 – Anna: Well… I finally got the last word.
30 – Siri: OK. See you soon.
31 – Neutral: Goodbye.

Table 5. Results of the user-oriented evaluation.

Question	Anna	%	Siri	%	Neutral	%
(1) Human like	28	93	21	70	11	37
(2) Interesting dialog	25	83	13	43	9	30
(3) Enjoy dialog	24	80	18	60	14	47
(4) Appropriate emotions	23	77	12	40	8	27
(5) Describe feelings	~	~	~	~	~	~
(6) Friend	16	53	8	27	6	20
(7) Best agent	15	50	9	30	6	20

The artificial agent appears more human-like than a neutral character and with a higher user engagement.

Concerning the emotion selection, in the majority of cases, it was able to adapt the emotional behavior according to the conversation. Despite the use of very different personalities, the character appears globally coherent.

However in some cases, the answers made by our system were not relevant. Here is one typical example. Note that this answer is different than the one reported in Table 4, since it occurred in another conversation context:

User: What is an artificial intelligence?
Anna: Have you lost faith in yourself?

This problem has potentially three causes. The first one is a bad design or malfunction of the selected agent, which provides an irrelevant response.

The second one is a problem of the Emotional Selection, which chooses an agent not compatible with the current emotional state. This may occur in rare situations since the roulette wheel selection has a low probability of choosing low weighted agents, but there is still a chance that it may happen by principle. However, note that the user does not always interpret such an example as a malfunction since it is a common human behavior to change the subject of the conversation or to make suboptimal responses.

The third potential cause is that the 12 available agents do not provide a complete and homogeneous coverage of the emotion space as shown in Fig. 8. There are two main approaches in order to solve this problem. The first one is to design emotion-based agents that perfectly fit the three-dimensional space. At least, an agent for each cube's edges plus one neutral agent at the center must be developed. The second solution is to increase the number of agents by designing much more

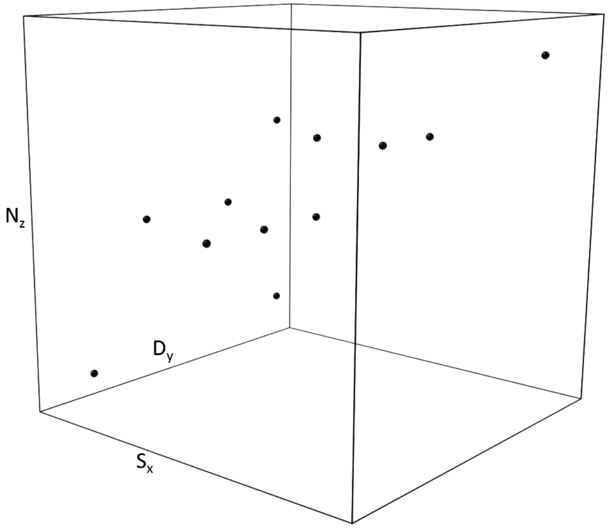


Fig. 8. Repartition of the 12 agents in the Lövheim Cube of emotions showing that they don't provide a full coverage of the three-dimensional space.

personality traits. This is our preferred solution since our goal is to obtain an "unforgettable" character. Also note that these two solutions are not mutually exclusive.

7 Conclusion

We have presented a multi-personality architecture with emotion selection for intelligent conversational agents. Our first experiments show that this approach is promising in terms of user engagement compared to a more neutral approach. It has shown also that despite the heterogeneity of the personality agents, emotion selection enables a globally coherent and believable character to emerge from conversations.

Of course, there are many works and studies that remain to be done. First of all, we need to design additional personality agents in order to have a better coverage of the three-dimensional emotion space. Secondly, we need to plan experiments involving much more participants, since 30 people reveal only an indication of a possible confirmation of our hypotheses. This will enable us to confirm these first results with both qualitative and quantitative evaluations of user engagement. Thirdly, we want to formalize our "edge of chaos" hypothesis and confirm it by testing different selection principles and algorithms.

References

1. Myers, K., Berry, P., Blythe, J., Conley, K., Gervasio, M., McGuinness, D., Morley, D., Pfeffer, A., Pollack, M., Tambe, M.: An intelligent personal assistant for task and time management. AI Mag. **28**(2), 47–61 (2007)
2. Heck, L.: Anticipating more from Cortana. Microsoft Research. http://research.microsoft.com/en-us/news/features/cortana-041614.aspx. Accessed 15 June 2017
3. Guha, R., Gupta, V., Raghunathan, V., Srikant, R.: User modeling for a personal assistant. In: Proceedings of the 8th ACM Web Search and Data Mining International Conference, Shanghai, China (2015)
4. Marcus, D.: Introducing Facebook M. https://www.facebook.com/Davemarcus/posts/. Accessed 15 June 2017
5. Weizenbaum, J.: ELIZA - a computer program for the study of natural language communication between man and machine. Commun. ACM **9**(1), 36–45 (1966)
6. Salovey, P., Mayer, J.D.: Emotional intelligence. Imagin. Cogn. Personal. **9**, 185–211 (1990)
7. Ekman, P.: Basic emotions. In: Dalgleish, T., Power, M. (eds.) Handbook of Cognition and Emotion. Wiley, Sussex (1999)
8. Bates, J.: The role of emotion in believable agents. Commun. ACM **37**(7), 122–125 (1994)
9. Dylaba, P., Ptaszynski, M., Maciejewski, J., Takahashi, M., Rzepka, R., Araki, K.: Multiagent system for joke generation: humor and emotions combined in human-agent conversation. J. Ambient Intell. Smart Environ. **2**(1), 31–48 (2010)
10. Leite, I., Pereira, A., Martinho, C., Paiva, A.: Are emotional robots more fun to play with? In: Proceedings of 17th IEEE Robot and Human Interactive Communication, Munich, Germany, pp. 77–82 (2008)
11. Hayes-Roth, B., Maldonado, H., Moraes, M.: Designing for diversity: multi-cultural characters for a multi-cultural world. In: Proceedings of IMAGINA 2002, Monte Carlo, Monaco, pp. 207–225 (2002)

12. Reeves, B., Clifford Nass, C.: The Media Equation: How People Treat Computers, Televisions, and New Media Like Real People and Places. CSLI Publications, Stanford (1996)
13. Seger, L.: Creating Unforgettable Characters. Henry Holt, New York (1990)
14. http://www.dictionary.com/browse/suspension-of-disbelief. Accessed 15 June 2017
15. Coleridge, S.T.: Biographia Literaria, Chap. XIV (1817)
16. McCrae, R.R., John, O.P.: An introduction to the five factor model and its Aplications. J. Pers. **60**(2), 171–215 (1992)
17. Heudin, J.-C.: A schizophrenic approach for intelligent conversational agent. In: Proceedings of the 3rd International ICAART Conference on Agents and Artificial Intelligence, Roma, Italy, pp. 251–256. Scitepress (2011)
18. Wolfram, S.: Universality and complexity in cellular automata. Phys. D Non Linear Phenom. **10**(1–2), 1–35 (1984)
19. Langton, C.G.: Computation at the edge of chaos: phase transitions and emergent computation. Phys. D Non Linear Phenom. **42**(1–3), 12–37 (1990)
20. Magnier, M., Lattaud, C., Heudin, J.-C.: Complexity classes in the two-dimensional life cellular automata subspace. Complex Syst. **11**, 419–436 (1997)
21. Langton, C.G.: Life at the edge of chaos. In: Artificial Life II, Santa Fe Institute Studies in the Sciences of Complexity, Redwood City, CA, p. 76. Addison-Wesley (1992)
22. Heudin, J.-C.: Emotion selection in a multi-personality conversational agent. In: Proceedings of the 9th International ICAART Conference on Agents and Artificial Intelligence, Porto, Portugal (2017)
23. Lungu, V., Florea, A.M.: Artificial emotion simulation techniques for intelligent virtual characters. Research Report of the Department of Computer Science, Polytechnics University of Bucharest (2012)
24. Gebhard, P.: ALMA - a layered model of affect. In: Proceedings of the 4th ACM International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 29–36 (2005)
25. Heudin, J.-C.: Evolutionary virtual agent. In: Proceedings of the IEEE/WIC/ACM Intelligent Agent Technology International Conference, Beijing, China, pp. 93–98 (2004)
26. Morris, W.N.: Mood: The Frame of Mind. Springer, New York (1989). https://doi.org/10.1007/978-1-4612-3648-1
27. Campos, J.J., Mumme, D.L., Kermoian, R., Campos, R.G.: A functionalist perspective on the nature of emotion. In: Fox, N.A. (ed.) The Development of Emotion Regulation, Monographs of the Society for Research in Child Development, vol. 59(2–3), pp. 284–303 (1994)
28. Heudin, J.-C.: A bio-inspired emotion engine in the living Mona Lisa. In: Proceedings of the ACM Virtual Reality International Conference, Laval, France (2015)
29. Mehrabian, A.: Pleasure-arousal-dominance: a general framework for describing and measuring individual differences in temperament. Current Psychol. **14**(2), 261–292 (1992)
30. Lövheim, H.: A new three-dimensional model for emotions and monoamine neurotransmitters. Med. Hypotheses **78**, 341–348 (2012)
31. Tomkins, S.S.: Affect Imagery Consciousness, vol. I–IV. Springer, New York (1991)
32. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the Second International Conference on Genetic Algorithms and their Applications, pp. 14–21. Lawrence Erlbaum Associates, Hillsdale (1987)
33. Linell, P.: Approaching Dialogue: Talk, Interaction and Contexts in Dialogical Perspectives. John Benjamins Publishing Company, Amsterdam (1998)

Towards General Cooperative Game Playing

João Marinheiro[1] and Henrique Lopes Cardoso[1,2(✉)]

[1] DEI, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal
`joca.mar2011@gmail.com`, `hlc@fe.up.pt`
[2] LIACC – Laboratório de Inteligência Artificial e Ciência de Computadores, Porto, Portugal

Abstract. Attempts to develop generic approaches to game playing have been around for several years in the field of Artificial Intelligence. However, games that involve explicit cooperation among otherwise competitive players – *cooperative negotiation games* – have not been addressed by current approaches. Yet, such games provide a much richer set of features, related with social aspects of interactions, which make them appealing for envisioning real-world applications. This work proposes a generic agent architecture – *Alpha* – to tackle cooperative negotiation games, combining elements such as search strategies, negotiation, opponent modeling and trust management. The architecture is then validated in the context of two different games that fall in this category – Diplomacy and Werewolves. Alpha agents are tested in several scenarios, against other state-of-the-art agents. Besides highlighting the promising performance of the agents, the role of each architectural component in each game is assessed.

Keywords: Multi-agent systems · Cooperative games
General game playing · Negotiation · Strategy · Opponent modeling

1 Introduction

From the beginning of AI research, games have been an important test-bed to develop new and interesting strategies and models for a variety of applications. While there has been extensive research using a number of different games, most work in this area relates to specific individual games. As a consequence, agents developed for one game are often difficult to adapt to other games due to the use of game-specific heuristics and architectures. One field of research that has attempted to mitigate this problem is that of general game playing [12,19]. General game playing agents provide useful insight into what are the essential elements for AI to emulate human thought and adapt to new situations never encountered before.

Most existing work in the field of game-playing agents relates to traditional adversarial games like Chess [9] or Go [20], because of their simple rules and large strategic depth. These kinds of games, while useful in many regards, do not

© Springer International Publishing AG, part of Springer Nature 2018
N. T. Nguyen et al. (Eds.): TCCI XXVIII, LNCS 10780, pp. 164–192, 2018.
https://doi.org/10.1007/978-3-319-78301-7_8

provide the best environment to allow modeling more complex and interesting social interactions between players. One interesting category of games that allows social interactions is that of *cooperative negotiation games*, where players are encouraged to barter and create or break deals between themselves in order to obtain the best results in the game.

Targeting general game playing AI for cooperative games would allow for the development of increasingly interesting and complex agent capabilities and features, enabling agents to not only adapt their playing strategies to different games but also negotiate in a variety of different environments with different protocols, goals and issues. In turn, this would allow to address increasingly complex and interesting real-world scenarios, a concern that should always be in place when doing research in games. To aspire such an aim, it is important to first determine the essential elements that agents must have in order to be able to play cooperative negotiation games effectively. Towards that direction, this work attempts to identify some of these elements and propose a generic agent architecture that tackles them and facilitates the development of agents with such capabilities, able to play cooperative negotiation games.

The rest of this paper is structured as follows. Section 2 provides insight into cooperative games, including their characteristics and challenges. In Sect. 3 we revise some of the existing approaches to generic game playing and to cooperative multi-agent games in particular. Section 4 introduces a general architecture for cooperative negotiation games, including a description of each of its modules, and a brief description of a general framework implementing the architecture. Then, in Sect. 5, we describe implementations of the meta-model described in Sect. 4, delivering some agents for two cooperative negotiation games. Section 6 reports on experimental evaluation of the developed agents, which have been tested against state-of-the-art agents, when available. Section 7 puts the contributions of this paper in the perspective of a long-term goal of delivering general cooperative game playing agents. Finally, in Sect. 8 conclusions are drawn and avenues of future work are laid out.

2 Cooperative Multi-agent Games

Traditionally in the field of game theory, a game is considered a cooperative game [15] if players are able to form binding commitments with each other. Games in which players cannot create binding agreements are then considered non-cooperative games. It is usually assumed that communication between players is allowed in cooperative games but not in non-cooperative games.

For the purposes of this work however, we consider a somewhat more general definition of cooperative games – games in which cooperation between players is possible and encouraged but in which binding agreements are not necessarily prevalent. More specifically, we will focus on cooperative negotiation games with a *mix of cooperation and competition*. In this setting, negotiation is used to establish cooperation in specific phases of an otherwise competitive game. Some characteristics that are frequently present in these games are:

- Very large search spaces, which makes the application of traditional search techniques impractical.
- Difficulty in evaluating moves and player positions due to the fact that evaluating such moves often depends on the *social context* of the game.
- The possibility of betrayals and desertions due to the existence of non-binding agreements.

There are many kinds of cooperative negotiation games, such as *Settlers of Catan*, *Quo Vadis?* or *Genoa*. In the context of this work, two very relevant cooperative negotiation games are *Diplomacy* and *Werewolves of Miller's Hollow*.

Diplomacy [3] is a strategy game for 7 players where each player takes control of a nation and their armies and navies. By submitting orders to these units, which are executed simultaneously with those of other nations, players attempt to capture territories and hold supply centers in a map of Europe. The first player to capture 18 or more supply centers wins the game. Players can also issue orders to support the orders of other players in the game, and they are encouraged to negotiate among each other in order to form alliances, support each other and create joint plans. Because of this the game highly encourages cooperation and players that are able to effectively negotiate are able to obtain much better results in the game.

Werewolves of Miller's Hollow [8] is a team-based game where two teams – the villagers and the werewolves – attempt to eliminate each other. The twist is that players on the villager team do not know who their allies are and who the werewolves are, encouraging players to communicate, share information and decide who to trust. The game is played in two phases: the day phase where players communicate freely, which ends with a vote to eliminate one player who the remaining players think might be a werewolf, and the night phase where players may not communicate but can choose to use some special abilities depending on their role in the game.

Both Diplomacy and Werewolves of Miller's Hollow are negotiation games with a mix of cooperation and competition that allow players to communicate among themselves in order to reach non-binding agreements, and use their actions to support or hinder each other. Both are deterministic and imperfect information games that work by phases. Despite these similarities, however, they are games with very different features. While Diplomacy is for the most part a zero-sum game, Werewolves of Miller's Hollow is not, with several players often losing or gaining utility with certain actions. While Diplomacy is a competitive game by nature where each player is ultimately hoping to be the one to win the game itself, which encourages eventual betrayal even among long time allies, Werewolves of Miller's Hollow is a cooperative game where players are divided into teams and encouraged to cooperate among themselves – agents win or lose the game as a group. The difference to a typical team based game is that in Werewolves of Miller's Hollow players do not have complete information about who is on their team and who is on the opposing team, and must thus be cautious about who they choose to trust. Table 1 summarizes a comparison of the characteristics of Diplomacy and Werewolves of Miller's Hollow.

Table 1. Comparison between Diplomacy and Werewolves of Miller's Hollow

Feature	Diplomacy	Werewolves
Deterministic	Yes	Yes
Information	Imperfect	Imperfect
Zero-Sum	Yes	No
Non-binding agreements possible	Yes	Yes
Simultaneous moves	Yes	Yes
Communication	Public & Private	Mostly public
Player victory	Individual	Team

Due to the characteristics of cooperative negotiation games, it is possible to obtain much better results in these games if one is able to negotiate and coordinate with other players effectively. Unfortunately, while humans are very good at negotiation and intuitively know who to trust, it is much harder for a computer to do so. In order to develop effective and believable AIs for this sort of games, and cope with the large size of their search spaces, new strategies that can effectively combine search strategies with negotiation and opponent modeling need to be employed.

3 Related Work

There are relevant works both in the area of general game playing and cooperative negotiation games. We here do not aim to provide an exhaustive list of game playing agents, but instead to discuss some of the most important approaches to general game playing and to cooperative negotiation games.

There have been several attempts to develop generic game playing agents that are able to understand and play a variety of games. These systems usually require a set of rules and constraints that formalize how the game is played, usually defined in a specific game description language.

Zillions of Games is a system developed by Mallet and Leffer [4], where programmers can define a wide variety of two dimensional abstract board games using rule files written in a proprietary language (ZRF files). The platform can then read these files and generate the game as well as intelligent general AIs that can play it. While the system is limited in the rules and layouts of the games it can generate (and its AI lacks advanced features such as negotiation or trust reasoning capabilities), it is nevertheless an interesting example of a general AI that is adaptable enough to play in a variety of different environments.

One of the most well known projects in this field is Stanford University's General Game Playing (GGP) project [12,19]. This project provides a framework upon which developers can create general game playing agents to play a variety of games, as well as the tools to describe those games. Games are described in a description language called GDL, which is then interpreted by the agents developed using the GGP framework. While GGP is mostly focused towards traditional board games a similar project by the University of Essex, the General Video Game AI (GVGAI), is more focused towards computer video games [17]. Similarly to GGP it provides a game description language, VGDL, and a framework upon which agents can be built, that are able to interpret and play any video game described using that language.

An approach to handle a particular family of imperfect information games is the Poker Game Description Language (PGDL) [5], focused in Poker and its many variants. PGDL is a language that allows users to define any Poker variant. Additionally, the PokerLang [18] high-level language facilitates the specification of agent strategies and tactics for specifying Poker playing agents.

Finally, more recently, Google's DeepMind project aims to apply deep learning techniques to a variety of scenarios, including games. Using neural networks, agents have been created that are able to effectively play games which contain extremely large search spaces, such as Go [20]. This deep learning approach can be applied to a variety of games and scenarios as long as one has access to quality training data with which to train the agent.

In the area of cooperative negotiation games there have also been several agents developed that provide an interesting starting point. Most of the work in this area is focused on the classic game of Diplomacy. Unfortunately, while there exist many agents for Diplomacy, no existing approaches have been found for the game of Werewolves of Miller's Hollow.

DumbBot is one of the simplest existing Diplomacy playing agents and was developed by Norman [16]. This bot has no negotiation capabilities and uses a simple heuristic to decide its actions by preferring to choose moves that weaken its strongest opponents. DumbBot assigns a value to each territory that depends on who controls it and what units are around it, and then it assigns actions to every unit depending on those score values. While the method used is very simple, DumbBot obtains fairly good results and is frequently used as a benchmark for other Diplomacy agents.

One of the most important and influential negotiating agents is the Israeli Diplomat. The architecture of the Israeli Diplomat was designed to be a general negotiation architecture to be applied in a variety of situations. This architecture was used to create a Diplomacy playing agent [13]. The Israeli Diplomat tries to mimic the structure of a war-time nation. It consists of several components working together to choose the best course of action. These components are the Prime Minister, the Ministry of Defense, the Foreign Office, the Military Headquarters, Intelligence and the Strategies Finder. The diplomat keeps a knowledge-base of the relations it believes each nation has with each other as well as any agreements it has, its intention to keep them and its trust that others will keep them. This knowledge-base is updated by the different modules as the game progresses, and affects every decision that the diplomat takes.

Another interesting approach is that of D-Brane [6], an agent developed by de Jonge that makes use of the NB^3 algorithm [7] as well as a complex strategical module to find the best sets of moves to negotiate and play. NB^3 is based on the branch-and-bound search algorithm and mixes negotiation with search, so that the former can direct the latter. It was designed to be used in environments where the search space is too large for traditional search techniques to be employed, and where better solutions often require cooperation among several agents. D-Brane uses a basic form of opponent modeling by using the utility values of deals previously proposed and accepted by its opponents as a way to direct the search for better solutions; however, it does not make an attempt to explicitly predict an opponent's goals or strategies. Another aspect that this agent lacks is the ability to negotiate coalitions with other agents as well as joint moves for future phases of the game, having no negotiation strategy for these kinds of deals.

DipBlue [10] is another negotiating agent for Diplomacy, inspired by the Israeli Diplomat architecture. DipBlue is split into several modules called Advisers, that together decide the actions the agent takes. Each adviser receives the evaluated move scores from previous advisers and alters them according to its role. The base adviser is inspired by DumbBot and uses the same scoring heuristic. This score is then changed by other advisers to promote support actions for the units, promote actions that keep agreements with its allies and encourage the agent to attack players that it distrusts. In order to model the trust value of each player, DipBlue keeps a trust matrix that is updated as the game is played [11]. If a player performs hostile actions against DipBlue, such as attacking it or breaking an agreement, its trust value diminishes. If a player performs friendly actions, or refrains from doing hostile actions, its trust value increases. DipBlue is more likely to accept agreements and help players with which it has a high trust value, and attack players with a low trust value. DipBlue does not have full negotiation capabilities, lacking the ability to ask and give information or threaten players.

Concerning other cooperative negotiation games, perhaps one of the more interesting approaches is the work by Afiouni and Øvrelid [1] that builds upon the opponent modeling techniques described by Krimpen *et al.* In their work, they propose a negotiating agent that uses weighted constraints to evaluate offers. By watching the variation in issues in offers proposed by its opponents, this agent can add or remove constraints from its opponent model, or alter their weights. It then proposes new offers by solving a prioritized constraint satisfaction problem.

4 A General Architecture for Cooperative Negotiation Games

A general architecture for cooperative negotiation games is needed to address the goal of obtaining general cooperative game playing. With this in mind, in this section we describe the *Alpha architecture* (first introduced in [14]) and its modules. This architecture enables approaching several of the challenges present in cooperative negotiation games, and facilitates the development of agents to play these games effectively.

4.1 The Alpha Architecture

In order to address the development of agents for cooperative negotiation games, we need to take into consideration several complex issues, such as negotiation

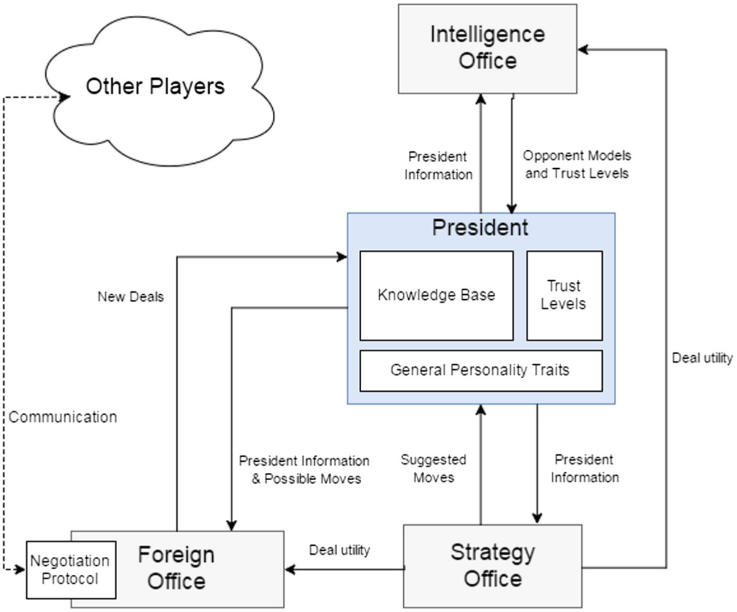


Fig. 1. High-level diagram of the Alpha architecture and its modules.

abilities, opponent modeling and trust relationships. Making the architecture generic and game-independent will ensure its applicability in a wide range of cooperative negotiation games. As such, no assumptions should be made regarding the usage of specific negotiation protocols or strategies, and the formalization of goals and search strategies should also be game-agnostic. The *Alpha architecture* is modular and based on the architectures of Israeli Diplomat [13] and DipBlue [11]. Figure 1 shows a simplified overview of the Alpha architecture.

The architecture has a structure similar to the Israeli Diplomat, with four independent modules:

- The **President** coordinates the interactions between all modules and is responsible for the final decisions regarding game play.
- The **Strategy Office** is in charge of suggesting good strategies to the President.
- The **Foreign Office** deals with negotiation with other players in the game.
- The **Intelligence Office** makes predictions regarding what opponents are likely to do in the game.

This structure allows the architecture to have a clean separation between different independent modules that deal with different issues: negotiation, opponent modeling, strategic/tactical evaluation of the game, which are then combined with high-level agent personality and overall strategy. The idea is to enable and facilitate compositionality when building agents with specific characteristics, by combining these modules in different ways.

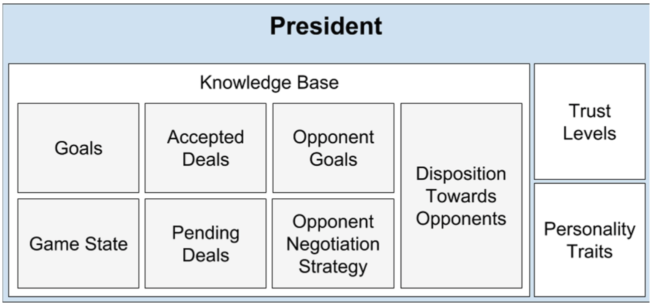


Fig. 2. High-level diagram of the President's components.

The President. The agent's central module is the President (PR), which holds its personality characteristics. This module coordinates interaction with the other modules, defining the overall high-level strategy of the agent through the definition of its goals. The PR is in charge of selecting and executing the agent's moves in the game. Figure 2 shows a simplified overview of the components of the PR module.

The PR keeps a knowledge-base including information about the game environment and its opponents. This knowledge-base is used and modified by the remaining three modules, allowing the PR to make decisions with up-to-date information regarding the environment. The information contained in the knowledge-base is the following:

- The current state of the game.
- The moves played during the course of the game by each player.
- The agent's current goals and their importance in the game.
- The lists of confirmed, completed and proposed deals by the agent and other players over the course of the game.
- The player's current disposition towards other players, such as who are its allies and enemies.
- The opponent models for each other player.
- The general trustworthiness levels of each player.
- The trustworthiness levels of each deal.

Since the different modules share and use a lot of the same information, one design concern was simplifying the sharing and exchange of information among the different modules, which lead to the decision to include this central knowledge-base in the PR module, which is easily accessible to all other modules. This also allows for the President to override decisions taken by its subordinate modules, such as which deals are confirmed.

Additionally, the PR includes a set of personality traits that can be defined depending on the game being played. These govern the general strategy of the PR, such as how aggressive it is, how trusting of other players it is or how prone to taking risks it is. Finally, the PR also keeps lists of moves and deals suggested by the other modules, which the PR can then choose to execute depending on several factors.

When defining the PR module, the developer must define what constitutes a deal, a move and a goal, as these are game-specific concepts. Different games have very different possibilities for what moves and deals a player can make. Instances of these notions specify the game-specific elements that are stored in the knowledge-base and used by all modules.

One important role of the President is deciding what overall goals the agent is striving for and what relative importance to attribute to each goal. This information is passed on to other modules, which take it into consideration in their reasoning processes. This approach allows the PR to dictate the overall strategy it wants to follow to its subordinate modules, allowing them to focus on the individual details of what actions are more likely to be effective in attaining these goals. Different PR modules allow the developer to customize the agent's general strategy and personality, allowing for different player archetypes.

The Strategy Office. A strategical assessment of the game is conducted by the Strategy Office (SO), which has the responsibility of suggesting appropriate moves to the PR. The SO contains most of the game-specific heuristics, and evaluates the utility of possible moves and deals. For this reason, the SO is highly dependent and adapted to the specific game being played. It also defines the search strategy used to explore the space of possible moves.

The SO is conceptually split into two parts: the *search strategy* and the *evaluation method*. The search strategy is used for exploring the space of possible moves, while the evaluation method is used to assess the utility of moves and deals. While the search strategy can, in principle, be applied to different environments with a low amount of effort, tactical evaluation is, in general, entirely dependent on the game being played, as it relies on specific knowledge about the game's rules.

In order to find the best moves, the SO has access to the PR's knowledge-base, which includes information that is relevant to evaluate the utility of different moves and deals. The PR requests move suggestions to the SO, which replies with moves that are then stored in the PR's internal list.

Changing the SO amounts to choosing among different search strategies and heuristics for the game, which can have a major impact on a player's effectiveness.

The PR and SO modules were conceived because of a design concern with separating the long term "macro" strategy and the short term "micro" strategy. This separation, in the form of the PR and SO (and also the FO when it concerns negotiation), is useful since it allows for the agent to tackle complex problems more easily, by dividing them into smaller problems. The SO and FO only need to worry about a small subset of the overall game at a time, which simplifies the development of these agents, while also allowing the agent to be formulated through long term plans without having to concern itself directly with more minor details of how to execute them via the PR.

The Foreign Office. The purpose of the Foreign Office (FO) is to manage any interaction with other players and negotiating deals and coalitions in a way that best allows the PR to execute the moves it is considering. By sending a list of moves, the PR requests the FO to find supporting deals through negotiation with other players. Also using any other information available in the PR's knowledge-base, the FO autonomously communicates with other players and decides what deals to propose, reject and accept. When a deal is proposed, confirmed or completed the FO informs the PR so that these deals are appropriately stored in its knowledge-base.

The FO includes a Negotiation Strategy that determines what deals are proposed and accepted and what concessions the agent is willing to make. This module also defines the negotiation protocol used by the agent when communicating with other players. The decision of what protocol to use is often dependent on the game being played or even the specific development framework on top of which the agent is being implemented.

By changing the FO, a developer can customize the negotiation capabilities of the agent, allowing the use of different negotiation and concession strategies. Omitting this module altogether leads to having an agent with no negotiation capabilities.

Having all negotiation handled by the FO and all actions performed by the PR allows the architecture to be cleaner, by compartmentalizing all platform specific code for interacting with the game environment and other agents in these two modules. It also more easily allows the agent to concurrently negotiate agreements with its peers, while simultaneously considering and possibly executing other incompatible moves and deals.

The Intelligence Office. Collecting information about and building models of opponents is an important aspect of games, and particularly in those involving social interactions. The purpose of the Intelligence Office (IO) is to address these needs by calculating trust values and building opponent models for the different players in the game.

The IO is divided into two parts: the opponent modeling function and the trust reasoning function. The opponent modeling function outputs the predicted goals and their relative importance for each opponent, to be updated in the PR's knowledge-base. How this is done is often specific to each game, since the goals themselves as well as the actions and deals being analyzed are also game-specific. Similarly, the trust reasoning function outputs trustworthiness values both for each opponent as well as for each individual deal, depending on how likely they are to be kept.

Configuring the IO can allow the developer to customize the opponent modeling and trust reasoning strategies, or lack thereof, of an agent. This module is especially useful in conjunction with the FO, since negotiations are likely to benefit from a good opponent model and accurate trust reasoning.

The design decision of separating opponent modeling and trust reasoning in the IO is related to the fact that while both subjects deal with predicting what actions an opponent will do, they are independent of each other. For example, one can define an agent with no trust reasoning capabilities if this agent is not expected to negotiate, since trust reasoning is especially important for negotiation, while still using opponent modeling. Opponent modeling is useful regardless of whether negotiation is possible, since predicting what goals enemy players might have can have an impact on the tactics and strategy of the SO and PR.

4.2 The Alpha Framework

In order to facilitate the application of the Alpha architecture, a simple Java framework was developed, composed of several abstract classes representing each of the described modules and their behavior. These classes define what each module should do as well as the data that they can access and how this data is updated and communicated to and by each of the modules. Each module is defined in its own class and implements a specific interface available to every other module. In addition, there are several data classes that contain data relative to the game being played and the agent itself, such as the knowledge-base or the agent's personality traits.

In order to implement an agent using the Alpha framework, a developer has to implement the abstract classes of the modules he wishes to use. When implementing the modules, the developer must implement their abstract methods in order to define the domain-specific negotiation strategies, protocols, heuristics, models and message handling. In the PR, the developer defines the high-level strategy for the agent, such as how it decides which goals are more important and what disposition it has towards other players. Optionally, the developer can also specify actions for the agent to take before and after playing, such as initializing or cleaning up data. In the SO, the developer implements utility functions for moves and deals, as well as the search strategy used to find and suggest moves to the PR. In the FO, the negotiation protocol and strategies are implemented as well as how the agent sends and receives domain-specific negotiation messages. Finally, in the IO there are functions where the developer may implement trust reasoning and opponent modeling strategies.

Data produced by the different modules is automatically updated and made available to the relevant modules. To make use of such data (e.g. use opponent goal predictions, calculated by the IO, in order to enhance negotiations in the FO), a developer has access to it in the PR's knowledge-base.

The developer must also implement data classes with domain-specific definitions of moves, deals, goals and opponents. Having all of this set up, the different modules can be attached to the PR and the agent be made to play the game.

This framework is publicly available at: https://github.com/jocamar/Alpha-Architecture.

5 Building Agents with the Alpha Architecture

We here describe our experience on making use of the Alpha architecture, by implementing two agents for two very different cooperative negotiation games: Diplomacy and Werewolves of Miller's Hollow. These agents, developed for such different games, can be seen as proof of concept to test the Alpha architecture in distinct scenarios, each with its own challenges.

5.1 AlphaDip

AlphaDip is a Diplomacy playing agent heavily based on D-Brane [6], using a modified version of its strategic module as well as the NB^3 algorithm to search for the best moves. It has a few key improvements compared with D-Brane, the most notable ones being an improved strategic module, a defined strategy for negotiating coalitions and some ability to predict opponent goals and trustworthiness. A high-level diagram showing how the Alpha architecture was applied when building AlphaDip is shown in Fig. 3.

The President. As detailed previously, the agent's PR is in charge of dealing with the high-level strategy for the agent and holds a variety of information such as player goals, moves to execute and trust levels. In order to discuss AlphaDip's implementation it is necessary to understand how these concepts are represented by the PR, and consequently, understood by the remaining modules.

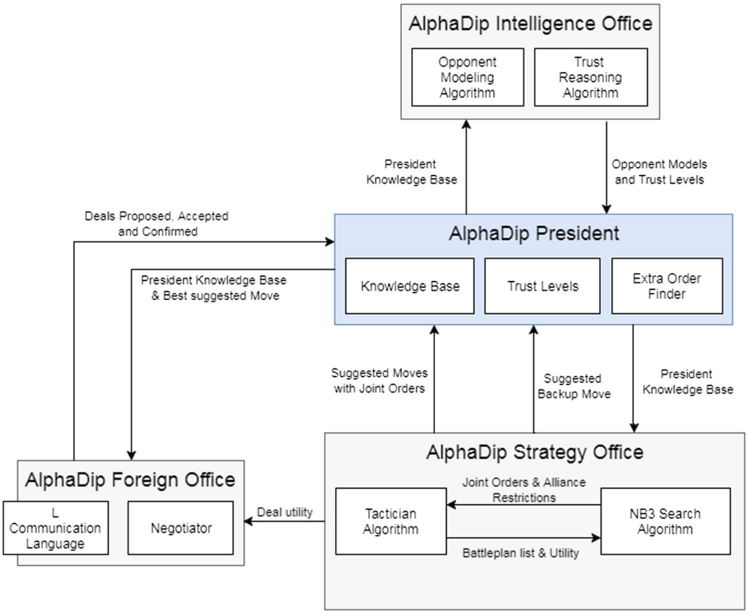

AlphaDip Intelligence Office
Opponent Modeling Algorithm
Trust Reasoning Algorithm
President Knowledge Base
Opponent Models and Trust Levels
Deals Proposed, Accepted and Confirmed
AlphaDip President
President Knowledge Base & Best suggested Move
Knowledge Base
Trust Levels
Extra Order Finder
Suggested Moves with Joint Orders
Suggested Backup Move
President Knowledge Base
AlphaDip Strategy Office
AlphaDip Foreign Office
Joint Orders & Alliance Restrictions
L Communication Language
Negotiator
Deal utility
Tactician Algorithm
NB3 Search Algorithm
Battleplan list & Utility

Fig. 3. AlphaDip's architecture.

In the context of AlphaDip, a move is a set of orders, one for each unit the player controls. In Diplomacy, winning the game amounts to having the goal of capturing as many supply centers as possible – this translates into modeling a player's goals as how much they want to control each supply center in the current game stage. We model this as positive or null real numbers, where 0 means no intention to control a supply center, 1 means neutral intention to control a supply center and greater values mean greater intentions of controlling a supply center. On the other hand, trust values for players stored by the PR are positive real numbers that are inversely proportional to the trustworthiness of these players: 0 represents full trust in a player, 1 represents neutral trust and greater values indicate lower levels of trust in a player.

The PR's main role is coordinating the remaining modules and executing moves. To fulfill this role, the PR first asks the SO for a fall-back move When a round starts: this is a move that is expected to work even without any supporting deals from other players nor any kind of negotiation, and will be used by the PR in case all negotiations fail, or in the absence of a FO. Afterwards, the PR periodically asks the SO, who continually searches for the best moves, for suggested moves to consider. As these moves are discovered by the SO, the PR forwards the last, and currently best known, move on to the FO to negotiate for any required deals. After a certain time has elapsed, if all of the prerequisite deals have been confirmed by the FO the PR selects the last move suggested by the SO. Otherwise it falls back onto the fallback move calculated at the start.

The Strategy Office. AlphaDip's SO tries to find moves that maximize the number of controlled supply centers, and is based on D-Brane's strategic module and the NB^3 algorithm. The objective of the game is to take control of as many supply centers as possible. As such, one way of determining the utility for a move is simply the number of supply centers that are ensured to be controlled by a player when it plays that move. This is how D-Brane calculates the utility of a move. AlphaDip calculates utility in a similar way, but introduces trust reasoning and the prediction of opponent goals in order to attempt to obtain a more accurate value than D-Brane. While D-Brane attributes the same score of 1 to every supply center, AlphaDip uses its goals to influence the value of each supply center. Additionally, if a move requires supporting orders from other players to succeed, their trust values are taken into account when determining the utility of the move. Using these computations, the SO suggests moves that are likely to be easy for the FO to obtain any necessary supporting move commitments from other players. Equation 1 shows how the SO determines the utility $U_p(m)$ of a move m for player p, where n is the total number of supply centers. Function I_m returns 1 if the supply center i is sure to be controlled after move m and 0 if not. Function g_p is the goal value that player p has or is assumed to have for each supply center. Finally, t_p is the average trust that player p has on all other players involved in the move or 1 if no other players are involved.

$$U_p(m) = \frac{\sum_{i=1}^{n} I_m(i) \times g_p(i)}{t_p(m)} \tag{1}$$

In order to find the best moves in a given round, the SO is divided into two components: the *Tactician* and the *Searcher*. The Tactician attempts to find the best set of orders taking into account certain constraints such as any existing order commitments. It then calculates the utility of the move as described above. The Tactician uses a similar method to D-Brane's strategic module [6]. This method, exemplified in Algorithm 1, splits the current round into several smaller battles for individual supply centers. For a given set of orders that attempt to capture a supply center – a battle-plan – it is simple to calculate whether the supply center will be captured. We do this by comparing it with every possible enemy battle-plan for that supply center (lines 9 and 11 in Algorithm 1). If a battle-plan ensures the capture or defense of a supply center, we say that it is an invincible battle-plan. We can also determine pairs of invincible battle-plans, that is, two battle-plans that if executed simultaneously guarantee that at least one of them succeeds (lines 16–19). After the Tactician has found the existing invincible battle-plans and pairs of battle-plans, an And-Or search is employed to find the largest set of compatible invincible battle-plans or invincible pairs, that are also compatible with any existing order commitments (line 25).

The Searcher uses the NB^3 algorithm to look for joint moves with other players that can maximize the utility for all players involved. Each node in the search tree is a joint battle-plan with one or more opponents that attempts to capture a supply center or help another player capture it. The path from a node in the tree to the root of the tree contains a set of commitments that the Tactician will attempt to solve for in order to find the best possible compatible orders for the remaining units and the subsequent utility value for that node. If an acceptable move is found, these commitments would then have to be negotiated by the FO with any other involved players, for the move to be able to be executed. By looking for joint moves, the SO can find strategies and moves that would not be possible if a player was acting completely independently.

Because the SO uses the trust values and predicted opponent goals when calculating the utility of a node, the search of the NB^3 algorithm will be directed towards nodes with joint moves with players in whom AlphaDip trusts, and who are more likely to accept the conditions of the joint commitments.

The Foreign Office. Since, in order to be successfully executed, the moves suggested by the SO may include order commitments with other players, the PR passes the current best move it is considering on to the FO for it to negotiate any required support agreements. AlphaDip's FO performs two types of negotiation: coalition establishment with other players and order commitments for the current round. This is an improvement over D-Brane, which did not have a strategy for the establishment of coalitions, instead assuming that all D-Branes simply formed a coalition against all other players in the game. Currently, AlphaDip is

Algorithm 1. Tactician Algorithm

```
playerAndOpponentBattleplans ← calculateBattleplans(gameState, player, allies)
playerAgreements ← getAgreementsFromPR()
for all $bp \in$ playerAndOpponentBattleplans do
    if !isCompatible($bp$,playerAgreements) then
        Remove $bp$ from playerAndOpponentBattleplans
    end if
end for
playerPlans ← getPlayerPlans(playerAndOpponentBattleplans)
opponentPlans ← getOpponentPlans(playerAndOpponentBattleplans)
for all $bp \in$ playerPlans do
    if !hasDefeatingPlans($bp$, opponentBattleplans) then
        Add $bp$ to invinciblePlans
    else
        defeatingPlans ← getDefeatingPlans($bp$, opponentBattleplans)
        for all $bp2 \in$ playerPlans do
            if isCompatible($bp$,$bp2$) then
                defeatingPlans2 ← getDefeatingPlans($bp2$, opponentBattleplans)
                if !hasLegalCombinationOfPlans(defeatingPlans,defeatingPlans2) then
                    Add $bp$ and $bp2$ as a new pair to invinciblePairs
                end if
            end if
        end for
    end if
end for
bestBattleplans ← getBestCombAndOrSearch(invinciblePlans,invinciblePairs)
return bestBattleplans
```

not able to negotiate move commitments for the following rounds as that would increase the complexity of the agent tremendously.

The strategy employed to negotiate coalitions is similar to DipBlue's [11]. At the start of the game, AlphaDip proposes a peace agreement to every other player in the game. After that, during the rest of the game the FO attempts to propose alliances against the stronger player in the game with which it is not in peace with. If a player's trust value rises above a certain threshold (meaning the player is less trusted) the peace with that player is broken. Conversely, if the trust value drops below a certain level (meaning the player is trusted) AlphaDip proposes peace to this player. Additionally, if the game has 4 or less players remaining, AlphaDip immediately breaks any alliances it has with a player if that player controls 14 or more supply centers. This is so that AlphaDip does not let a player get too close to winning in the final stages of the game.

The FO also attempts to negotiate joint order commitments for the current round. The PR periodically asks the FO to negotiate deals concerning the moves being currently considered. The FO compares the utility of the suggested moves with the utility of each of the proposals it received: the FO either accepts the best proposal received if it has more utility, or proposes any necessary joint order commitments required by the moves proposed by the PR.

In case the FO receives a proposal that is compatible with any deals it has already accepted, the FO asks the SO to calculate the utility of that deal, and informs the PR so that it stores the deal in the proposed deals list. The reason it does not choose to immediately accept or reject the proposal, as explained in [6], is so that the SO is given some time to continue searching and looking for any possibly better options for other joint moves, before the agent commits to the proposed orders. By committing to an offer and adding it to the PR's confirmed deals list, the search performed by the SO is automatically constrained to only look for moves that satisfy the conditions in the accepted deals.

The Intelligence Office. AlphaDip may use the IO to calculate trust values for players and predict their current goals in the game. In order to update the trust values, the IO uses a strategy similar to DipBlue [10], where trust in players increases steadily over the course of the game if no aggressive actions are taken by these players, and decreases when aggressive actions are taken. The magnitude of these updates is dependent on current trust values associated with the players, as well as whether AlphaDip considers himself to be at peace or at war with them. This way, if a player is highly trusted or in peace with AlphaDip, any aggressive actions it takes will have a bigger impact on that player's trust. On the other hand, if a player is not trusted or is at war with AlphaDip, any aggressive actions it takes have a smaller impact on that player's trust, since AlphaDip already expects that player to take aggressive actions.

The IO also attempts to predict its opponents' goals, that is, which supply centers it believes each player wants to control more, using a simple strategy exemplified in Algorithm 2. Each time a player takes an offensive action against a certain supply center, the IO increases the likelihood that that player wants to control that supply center (line 9 in Algorithm 2). If a player takes no offensive actions against a supply center, or takes actions that would help another player capture that supply center (such as support orders), the IO decreases the likelihood that the player wants to control that supply center (lines 5 and 12). Similarly to trust value updates, the magnitude of such increases and decreases are influenced by the current values for each supply center. This way, if a player is already expected to want control of a certain supply center, any actions it takes have a small impact on the value for that supply center desire. On the other hand, if a player suddenly makes a move on a supply center that AlphaDip believed that player was not interested in, the value for that supply center will be affected more significantly – this can be seen as an ability of AlphaDip in detecting changes in opponents' goals.

5.2 DipBlue

In addition to AlphaDip, we have made a reimplementation of DipBlue in light of the Alpha architecture (see Fig. 4). In order to do this we split the DipBlue

Algorithm 2. AlphaDip IO Goal Prediction Algorithm

```
playerOpponents ← getOpponentsFromPR()
opponentGoals ← getOpponentGoalsFromPR()
for all $op$ ∈ playerOpponents do
    for all $g$ ∈ opponentGoals[$op$] do
        $g \leftarrow g \times 0.99$
        actionsSupportingGoal ← getActionsSupportingGoal($op$,$g$)
        actionsAgainstGoal ← getActionsAgainstGoal($op$,$g$)
        for all $o$ ∈ ordersAndDealsSupportingGoal do
            $g \leftarrow g + \frac{0.04}{g}$
        end for
        for all $o$ ∈ ordersAndDealsAgainstGoal do
            $g \leftarrow g \times 0.95$
        end for
    end for
end for
return opponentGoals
```

architecture into SO, FO and IO. Our version of DipBlue works exactly the same as the original DipBlue described in [11].

DipBlue's advisers are part of the SO, and are used to calculate the utility value for possible moves in the same way as originally. Unlike with AlphaDip, in DipBlue each move is a single order for a unit, and each order has a utility value assigned by the advisers. After finding the best orders, the SO suggests them to the PR, who asks the FO to negotiate any deals it thinks are necessary. The FO implements DipBlue's negotiation strategy, requesting supports from its allies for any moves that could use them and negotiating alliances and peace deals.

The execution of DipBlue's moves is not dependent on the success of negotiations with other players, though any supports may increase the likeliness of those moves. As a result, the PR will always execute moves suggested by the SO, regardless of the result of any negotiations the FO attempts.

The IO updates opponent trust values in the same way the original DipBlue does. Unlike AlphaDip, DipBlue's IO does not predict opponent goals.

5.3 AlphaWolf

Werewolves of Miller's Hollow was also chosen to draw an implementation of the Alpha architecture. As far as we know, no frameworks are available to develop agents for this game. We have implemented a game server using the Jade multi-agent framework[1], for which we implemented an agent to play the game – AlphaWolf.

In order to simplify the implementation, and because certain roles are more suited to be played physically with humans, we use a simplified version of the game with a subset of the original player roles and abilities. In our version of the

---

[1] http://jade.tilab.com.

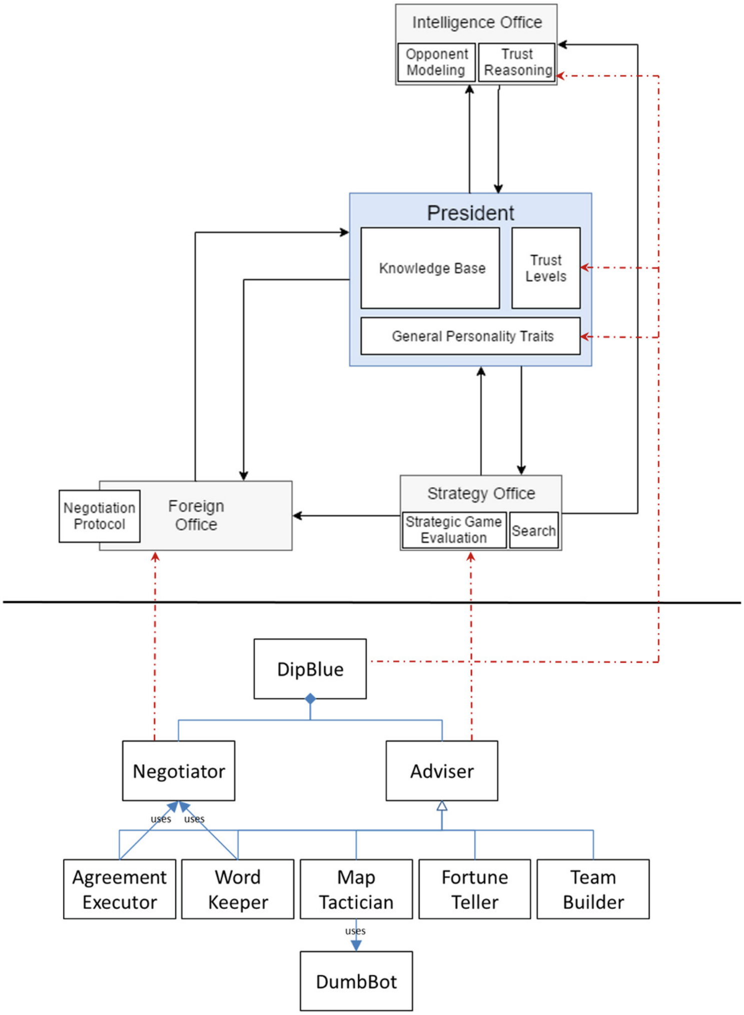


Fig. 4. Mapping DipBlue to the Alpha architecture.

game, there are 4 possible roles for the players: werewolves, villagers, seers and doctors. Werewolves have the goal of killing every other non-werewolf player in the game while every other player has the goal of killing the werewolves. The werewolves, seers and doctors each have a special ability that they can secretly perform during the night phase of the game. Werewolves can collectively vote on an enemy player to kill during the night. The seers can choose any player to investigate during the night, learning its secret role. Finally, doctors are able to choose a player, who if attacked by the werewolves during the night will be healed and remain in the game, informing the doctor that this happened.

The President. AlphaWolf's PR works as described in Sect. 4. At the start of each phase of the game it requests the IO to update opponent trust values and predicted goals. In the context of this game, a player's goal is tied to its role and as such predicting a player's goal means predicting the likeliness that a player has a certain role. In the PR's knowledge-base, this means that a probability is associated with each player-role pair. Role probabilities for a given player add up to 100%, so that if a certain role has a 100% probability, the PR knows that player's role and, consequently, its goal. A player's trust is represented by a positive or null real number, where 1 means neutral trustworthiness, 0 means no trustworthiness and values above 1 mean progressively higher trustworthiness.

The PR requests the SO to suggest a good move. The notion of move depends on the current phase of the game, but it always involves choosing a player to either vote out of the game or as a target for the player's ability during the night phase. After a player is suggested by the SO, and depending on whether the current phase of the game allows negotiation between the players, the PR may ask the FO to attempt to negotiate with the other players for joint votes against some player, or requests for investigation or healing. When the FO has finished negotiating, the PR decides to either vote or target a certain player for its special action. If no deal has been reached, the player is randomly chosen from the list of players suggested by the SO, with players with higher utility being more likely to be picked; otherwise, if a deal has been reached for a specific player, the PR will take the action it agreed to on the deal.

The Strategy Office. AlphaWolf's SO implements a simple strategy to suggest potential players to either attempt to eliminate or protect, depending on the current phase of the game and the player's role and goals. This is done by assigning each player a threat score, which is a measure of what roles and goals the player believes an opponent has (as calculated by the IO) and how dangerous these roles are to the player. In general, if a player is on the werewolf faction, roles that have the ability to gather more information or use abilities that hinder werewolves actions will have a higher threat level to the player. In the same way, if a player is on the villager faction, roles that have the ability to gather more information or hinder the werewolves are less likely to kill the player, and thus are less threatening. Depending on the current phase of the game and whether the actions available to the player will hinder an opponent (such as voting to kill it) or help another player (such as healing it), either this threat score is used as the utility for the move or its inverse is used, respectively. Equation 2 shows how the threat value T_p of player p is calculated, where n is the number of different possible roles a player can have, B_i is the base threat value for role i and C_i is the current estimated likelihood that player p has role i.

$$T_p = \frac{\sum_{i=1}^{n} B_i \times C_i}{\sum_{i=1}^{n} B_i} \tag{2}$$

The SO also has the purpose of calculating the utility of deal proposals. This utility is based on the previously mentioned threat value of the proposer of the deal, the threat value of the player whom the deal concerns, what type of action the deal is proposing and the trust of the player in the proposer of the deal. This calculation is described in Algorithm 3. The type of action proposed and the threat values for the proposer and the player affected by the proposal are used to calculate two values – one for the proposer and one for the target of the proposed action – representing how much the player is willing to help the proposer and hurt the target. These two values are then multiplied together with the trust on the proposer, representing how much the player trusts the proposer to abide by the deal and not take any actions against him, to reach the final utility value for the deal (line 9).

Algorithm 3. AlphaWolf SO Deal Utility Calculation

```
1: target ← getTargetFromDeal()
2: proposer ← getProposerFromDeal()
3: proposerValue ← $\frac{1}{getThreatValue(proposer)}$
4: if dealActionIsPositive() then
5:     targetValue ← $\frac{1}{getThreatValue(target)}$
6: else
7:     targetValue ← getThreatValue(target)
8: end if
9: return targetVales × proposerValue × getTrustFromPR(proposer)
```

The Foreign Office. In Werewolves of Miller's Hollow, players can only communicate during certain phases of the game, namely the discussion phase and, for werewolves, the night phase. As the game is very reliant on communication between players, negotiation is very important in order to obtain effective players. Otherwise, players would not be able to coordinate their votes or use their abilities during the night. This is the purpose of the FO.

AlphaWolf's FO implements a simple negotiation strategy during the day phase, where each player proposes a joint vote against another player who they think is the most threatening, as well as other agreements such as investigation or heal requests, depending on their levels of trust with other players. Players then wait for agreement confirmations from their opponents, locking the agreement in place if they receive a confirmation. In each negotiation round, players compare the utility of the proposals they receive with their concession value, which is based on their own proposal and decreases over time, and decide either to continue waiting or accept another proposal, retracting their own.

In the case of the nightly negotiation phase for werewolves (where they coordinate to choose a victim), the strategy employed by the FO can be even simpler, since the werewolves have complete information about who the other werewolves are and can thus assume that they are working towards the same goals. In this case, consensus is reached by means of a sealed bid mechanism, where each werewolf proposes one player to vote for as well as their preference level for that player. After all werewolves have made their proposals, the bids are counted and the player with the highest preference level among all werewolves is selected, with every werewolf voting for it.

The Intelligence Office. AlphaWolf's IO has the function of attempting to predict a player's goals and its trustworthiness. As mentioned previously, since a player's goals are tied to its role in the game, predicting its goals is a matter of predicting its role. In order to predict an opponent's role, the IO analyses the proposals and votes of that player over the course of the game. The predicted role probabilities for that opponent are thus a function of the threat values of the players that opponent voted against (or proposed to vote against), and the rounds in which that player took those actions. Algorithm 4 describes the calculation for the prediction of opponent goals by the IO.

Algorithm 4. AlphaWolf IO Role Prediction Calculation

```
 1: pastRounds ← getPastRoundsFromPR()
 2: opponents ← getOpponentsFromPR()
 3: opponentGoals ← getOpponentGoalsFromPR()
 4: for all $op$ ∈ opponents do
 5:     for all $round$ ∈ pastRounds do
 6:         roundAgeFactor ← getAgeFactor($round$)
 7:         for all $actions$ ∈ getOpponentActions($op$, $round$) do
 8:             target ← getTarget($action$)
 9:             voteDamage ← $\frac{getThreatValue(target)}{getAverageThreatValue()}$ × roundAgeFactor
10:             scalingFactors ← calculateRoleScalingFactors(voteDamage)
11:             for all $r$ ∈ opponentGoals[$op$] do
12:                 $r \leftarrow r\times$ scalingFactors[$r$]
13:             end for
14:             normalizeOpponentGoals(opponentGoals[$op$])
15:         end for
16:     end for
17: end for
18: return opponentGoals
```

The IO searches through each player's past actions and for each vote or proposal that player made it calculates a vote or proposal damage value (line 9 in Algorithm 4). This value indicates the likelihood that an action was taken with the intent of damaging the player's faction and is based on the threat values of the players who are the targets of that opponent's actions. A high threat value for the target of the action indicates that the action was not very damaging to AlphaWolf's faction, and even may have been helpful, and a low threat value indicates a damaging action, as that opponent was voting against players that are considered likely to be allies.

For each action, its vote damage is then used to calculate a scaling factor for each possible opponent role (line 10), which is finally used to scale the role probabilities of each role proportionally (lines 11–12). Each role has a different scaling factor because certain roles are more likely to have more information than others: seers have a higher scaling factor than doctors, and doctors have a higher scaling factor than villagers. In this way, if an opponent takes a damaging action, the likelihood that it is a seer comparatively diminishes more than the likelihood that it is a villager, since a seer would be less likely to commit damaging actions against the villagers, having more information about the player roles. These scaling factors are then multiplied with each current role probability, and the values for the roles are then re-scaled back so that they total 100% (line 14).

To calculate trust values, the IO analyses the previous round and checks for each opponent if it kept any agreements it accepted or if it voted against the player. If an opponent broke an agreement or voted against the player, its trust value is decreased by a certain amount, otherwise its trust value increases.

6 Experimental Validation and Evaluation

In order to validate our implementation of the Alpha-based agents described in Sect. 5, we have setup a number of experimental scenarios. Besides illustrating the correct functioning of the agents themselves, these scenarios also helped in evaluating individual architecture components and, when available, enabled us to compare the performance of the agents with other state-of-the-art players.

6.1 AlphaDip Validation Setup

We compared AlphaDip with three previously developed Diplomacy playing agents: DumbBot, DipBlue and D-Brane. For that, we performed tests similar to those reported in [6,10]. It should be noted that unlike the D-Brane tests reported in [6], which assumed D-Branes always formed a coalition against every other agent in the game, we allow our agents to negotiate at will, establishing and breaking coalitions.

In each experimental setup, we tested AlphaDip using 3 distinct configurations, in order to separately assess the impact of negotiation and opponent/trust modeling in its performance: (i) using only the PR and SO, (ii) including the FO, and (iii) using all four PR, SO, FO and IO.

For every configuration, in each experiment we played a number of games of Diplomacy, stopping after 40 game phases. After the end of a game, we ordered players by ranking, from 1st to 7th, and collected ranking results. Ranking is determined by the number of supply centers a player controls at the end of the game (in case it is still alive), or by the game phase in which the player was eliminated. Players with more supply centers or eliminated at later phases have a higher rank in the game and are thus considered better. For configurations without the FO (and thus, with no negotiation capabilities) we played a total of 100 games. For configurations including the FO we set the negotiation deadline at 15 s per round. Since these tests take considerably longer to execute, we only played a total of 50 games per configuration.

All tests and experiments were performed using a laptop computer with 8 GB of RAM and an Intel Core i5-6440HQ mobile CPU clocked at 3.5 GHz.

6.2 AlphaDip Results

Similar to experiments reported in Ferreira's [10] and de Jonge's [6] works, in each experiment we had two instances of AlphaDip playing against 5 instances of DumbBot. Combining the ranks of our AlphaDip agents, the best possible average rank is 1.5, while the worst possible average rank is 6.5. We can compare the average ranks of AlphaDip with the average ranks of DipBlue and D-Brane reported in [6,10], respectively. The best rank achieved by DipBlue in the tests performed by Ferreira is 3.57 [10], while the best rank obtained by D-Brane in the tests performed by de Jonge is 2.35 [6].

Table 2 shows the average rank obtained by AlphaDip in each configuration (standard deviation is also included). These results show that AlphaDip plays significantly better than the DumbBot and DipBlue, even without negotiation, opponent and trust modeling capabilities. However it also appears that the inclusion of IO and FO only has a small effect on the performance of the agent. A t-test performed on these results obtains a score of 0.554 when comparing the second with the third configuration, and a value of 0.109 when comparing the first with the third. This points towards statistically significant performance differences between the base configuration (PR+SO) and the full one (PR+SO+FO+IO), although further testing is required to confirm this hypothesis.

In order to test how well AlphaDip performs against other, more advanced, agents we also performed an experiment including two AlphaDips and two D-Branes (in this case without negotiation capabilities[2]), together with 3 DumbBots. The results of these tests are shown in Table 3. Similarly to the previous experiment, these tests show that AlphaDip outperforms D-Brane, especially when it can use the negotiation and opponent modeling capabilities provided by the FO and IO. When playing only with the SO, AlphaDip performs only slightly better than D-Brane. Because AlphaDip's SO is based on D-Brane's strategic module, however, this difference is not statistically significant.

Table 2. Average rank of 2 AlphaDip when playing with 5 DumbBot.

2xAlphaDip & 5xDumbBot	
AlphaDip config.	Avg. rank
PR + SO	2.35 ± 0.15
PR + SO + FO	2.21 ± 0.21
PR + SO + FO + IO	2.11 ± 0.19

---

[2] The public versions of D-Brane do not include negotiation capabilities (http://www.iiia.csic.es/~davedejonge/bandana).

Table 3. Average ranks in games with 2 AlphaDip, 2 D-Brane and 3 DumbBot.

2xAlphaDip, 2xD-Brane & 3xDumbBot		
AlphaDip config.	AlphaDips avg. rank	D-Branes avg. rank
PR + SO	2.84 ± 0.17	2.91 ± 0.16
PR + SO + FO	2.37 ± 0.21	3.19 ± 0.24
PR + SO + FO + IO	2.29 ± 0.20	3.25 ± 0.24

Table 4. Average ranks in games with 2 AlphaDip, 2 DipBlue and 3 DumbBot.

2xAlphaDip, 2xDipBlue & 3xDumbBot		
AlphaDip config.	AlphaDips Avg. rank	DipBlues Avg. rank
PR + SO	2.38 ± 0.16	5.03 ± 0.20
PR + SO + FO	3.56 ± 0.26	3.44 ± 0.32
PR + SO + FO + IO	2.3 ± 0.22	4.73 ± 0.29

In order to complement the previous experiments, we also tested AlphaDip in an environment with a higher number of negotiating agents: 2 AlphaDips play with 2 DipBlues and 3 DumbBots. The 2 DipBlues played with the standard adviser configuration described in [10]. Results are shown in Table 4.

These results show that when AlphaDips are playing with a PR+SO configuration or with all modules running they get a similar average rank, between 2.3 and 2.4. A statistical t-test gives us a value of approximately 0.653, which shows that the difference observed is not statistically significant. The inclusion of negotiation and trust reasoning does not seem to significantly affect the performance of AlphaDip in this experiment. On the down side, negotiation without opponent modeling seems to be counterproductive, as using the FO without the IO (meaning that AlphaDip negotiates without making any attempt to predict opponent goals or trustworthiness) brought the worst results in this experimental setup – the average rank decreases to 3.56.

This lack of impact from negotiation resembles results obtained by de Jonge in [6], where he points out that even though the NB^3 algorithm manages to find good joint moves (when they exist), their impact in the overall result of the game is negligible. Negotiating joint moves for only the current round is not enough to significantly increase the performance of the players – in order to obtain better results one would have to attempt to negotiate further rounds ahead as well.

Another interesting observation is that, while AlphaDips do obtain a small increase in their average rank when all modules are running as compared to having just PR+SO, DipBlues themselves also benefit from this setting: DipBlues obtain a better average ranking of 4.73 when playing against AlphaDips with all modules active, compared with an average ranking of 5.03 when playing with AlphaDips incapable of negotiation. These observations were also corroborated by de Jonge's observations in [6], where he found that other agents could also benefit from the deals discovered by agents running the NB^3 search algorithm.

6.3 AlphaWolf Validation Setup

Testing AlphaWolf is not as straightforward as testing AlphaDip, due to the lack of agents developed for this game. Nevertheless, we conducted experiments that allowed us to test the relative performance of our agent with different configurations of active modules.

In order to test the impact of each module in the performance of the agent, we opted to have AlphaWolf werewolf agents always playing with all modules running, and changed only the configurations of AlphaWolf villager agents. This way, we can easily see the effect each module has on the performance of villagers. As before, We tested 3 different configurations for this agent: (i) using the PR and the SO, (ii) including the FO, and (iii) using all four PR, SO, FO and IO.

In each test we had the agents play 100 games in a 10 player game where 2 of the players were werewolves, and the remaining 8 were from the villager faction, with 1 seer, 1 doctor and the remaining 6 being standard villagers. This ratio of werewolf to villager players was chosen because it is the recommended ratio in the official Werewolves of Miller's Hollow rules [8]. We recorded the win percentage for the villagers over those games as well as the average number of villager agents left alive at the end of the game when the villager faction won.

6.4 AlphaWolf Results

Unlike with Diplomacy, results for the AlphaWolf agents show that negotiation, trust and opponent modeling are effective features, with a significant impact on agent performance (see Table 5). With the inclusion of the FO and the IO, performance steadily increases from a 27% to a 46% win ratio, and finally to a 73% win ratio with all modules active. The inclusion of the IO also significantly increases the number of villagers remaining alive in games where this faction wins. This indicates that, with the inclusion of trust and opponent modeling, players are able to identify werewolves much earlier in the game, allowing for quicker victories.

One possible explanation for the difference in the relative impact of negotiation, trust and opponent modeling in the game of Werewolves of Miller's Hollow

Table 5. Win percentages and average number of remaining villagers for a team of 8 villagers playing against 2 werewolves.

Villagers config.	Win %	Avg. # villagers
PR + SO	27%	3.56
PR + SO + FO	46%	3.20
PR + SO + FO + IO	73%	4.88

as compared with Diplomacy is that in the latter agents already implicitly have an idea of their opponents' goals. Since in Diplomacy the rules of the game encourage players to capture supply centers, one can assume that other players will try to maximize their number of supply centers over the course of the game. On the other hand, in Werewolves of Miller's Hollow players have no way to know, at the initial stage of the game, what their opponents will be trying to do. This means that players have no way to predict an opponent's utility function at the start of the game, and must analyze its actions in order to predict the opponent's goal.

Negotiation may also have a greater impact in Werewolves of Miller's Hollow because each player only has a single vote to affect the round. Without coordination and organizing joint votes, players have a hard time completing their goals. In Diplomacy, players can have differing numbers of supply centers and units, which allows certain players to affect the outcome of the rounds more than others; this means that strong players can use their superior strength to obtain their objectives even without negotiation.

7 Discussion

While there have been several attempts to develop general game playing AIs, the task has not proven to be easy, and the available proposals often come with several important limitations. The path that has been taken throughout time is to address successively more and more complex characteristics of the environment faced by agents: from deterministic, sequential and perfect information games, to stochastic and imperfect information games. The next frontier may well be games demanding for social relationships, where negotiation and trust modeling are essential components to establish cooperation, a need that is magnified in the presence of simultaneous moves.

The low coverage of games that include this social element is one of the biggest limitations of current approaches such as Zillions of Games [4] and the General Game Playing project [19], which focus mostly on abstract board games involving no or little cooperation. The Alpha architecture seeks to bridge the gap between the current state-of-the-art and cooperative negotiation games.

However, it is important to note that the Alpha architecture and framework do not constitute in themselves a general game playing AI, as they do not offer some essential capabilities that other general game playing AIs possess. For example, a way for the developed agents to abstractly understand the rules of the different games is entirely missing (see [2] for a glimpse of the challenge). In order to obtain a truly general game playing AI it is necessary for the developed agents to be able to learn different rule sets for different games that the developers might not even themselves know, for example by reading a description language file (using formats such as [21] or [5]) that describes the rules for these games.

The proposed architecture is instead intended to serve as a strong framework, upon which game-specific agents can be implemented, that identifies, organizes and generalizes the use of a number of different capabilities required to do well at negotiation games. By using the proposed architecture, developers can easily implement agents that make use of a combination of search strategies with negotiation, opponent modeling and trust reasoning capabilities, in a general and standardized way. This basis can be used in the future to develop a true general game playing AI that is able to play a variety of cooperative negotiation games. This can be done by adding some improvements over the current work, while keeping the same role for each of the modules of the Alpha architecture.

8 Conclusions and Future Work

The objectives of this work were the study of what elements were necessary to create effective agents that could play cooperative negotiation games, and to develop a generic architecture including these elements, which could be used to facilitate the development of effective agents for a wide variety of games.

We tested the proposed architecture by developing agents for two very different cooperative negotiation games, and believe that the proposed Alpha architecture is generic enough to be applied to many other different games. The two most relevant agents developed using the Alpha architecture and framework were AlphaDip and AlphaWolf. AlphaDip is an agent with strategies inspired by D-Brane and DipBlue, with the inclusion of opponent modeling to make predictions about an opponent's intention to capture certain supply centers, as well as a negotiation strategy for the establishment of coalitions, which was not present in D-Brane. AlphaWolf is a Werewolves of Miller's Hollow agent that also includes negotiation, trust and opponent modeling capabilities, allowing it to predict its opponents' roles and negotiate deals accordingly.

The results of the tests performed using these two agents show that AlphaDip was in general superior to both DipBlue and D-Brane, obtaining better average ranks in the games played. However, the inclusion of negotiation, trust reasoning and opponent modeling capabilities did not have a very large impact on the performance of the agent. On the other hand, results obtained for AlphaWolf show that the inclusion of the Foreign Office and Intelligence Office had a larger impact in the performance of the agent. This indicates that negotiation, trust and opponent modeling are more important in Werewolves of Miller's Hollow than in the Diplomacy game.

We believe that these results are positive and the inclusion of negotiation, trust reasoning and opponent modeling capabilities generally improved the performance of the agents, though the impact was much greater for AlphaWolf than for AlphaDip. We also believe that the developed architecture and framework are a helpful contribution to the field by facilitating the development of agents with these capabilities. However, while the developed architecture is very modular and allows agents to be built upon it and make use of negotiation, trust and opponent modeling, there is lots of room for improvement. The Alpha architecture allows developers to define different negotiation strategies, trust reasoning and opponent modeling approaches, which may be tailored to a specific game. However, this process can be made simpler with the inclusion of generic strategies (such as D-Brane's NB^3 algorithm) that can be applied equally to any environment. The inclusion of a generic way to predict opponent goals and strategies, calculate trust values and decide what deals to accept, based on the knowledge-base of the President, would further simplify the process of developing an efficient agent.

The agents implemented during the course of this work, while generally efficient, could also be improved. One major improvement to AlphaDip is to allow the agent to search for and negotiate movement commitments for several rounds ahead, instead of only the current round. As for AlphaWolf and the implemented Werewolves of Miller's Hollow server, one key improvement would be the capability for AlphaWolf to use strategies involving bluffing, by for example making opponents believe it has a different role than its true role, a strategy human players frequently use. If correctly implemented, this ability could make AlphaWolf much more effective, especially when playing with human opponents.

References

1. Aouni, E.N., Øvrelid, L.J.: Negotiation for strategic video games. Master thesis, Norwegian University of Science and Technology (2013)
2. Bonnet, É., Saffidine, A.: On the complexity of general game playing. In: Cazenave, T., Winands, M.H.M., Björnsson, Y. (eds.) CGW 2014. CCIS, vol. 504, pp. 90–104. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-14923-3_7
3. Calhamer, A.B.: The Rules of Diplomacy, 4th edn. Avalon Hill, Baltimore (2000)
4. Zillions Development Corporation. Zillions of Games (2016). http://www.zillions-of-games.com. Accessed 16 May 2017
5. Correia, J.: PGDL: Sistema para definição genérica de jogos de Poker. Master thesis, Universidade do Porto (2013)
6. de Jonge, D.: Negotiations over large agreement spaces. Ph.D thesis, Universitat Autonòma de Barcelona (2015)
7. de Jonge, D., Sierra, C.: Negotiation based branch & bound and the negotiating salesmen problem. In: Proceedings of the 14th International Conference of the Catalan Association for Artificial Intelligence, Lleida, Catalonia, Spain (2011)
8. des Pallières, P., Marly, H.: Werewolves of Miller's Hollow: The Village. Lui-même (2009)
9. Drogoul, A.: When ants play chess (Or can strategies emerge from tactical behaviours?). In: Castelfranchi, C., Müller, J.-P. (eds.) MAAMAW 1993. LNCS, vol. 957, pp. 11–27. Springer, Heidelberg (1995). https://doi.org/10.1007/BFb0027053
10. Ferreira, A.: DipBlue: a diplomacy agent with strategic and trust reasoning. Master thesis, Universidade do Porto (2014)
11. Ferreira, A., Lopes Cardoso, H., Reis, L.P.: Strategic negotiation and trust in diplomacy – the DipBlue approach. In: Nguyen, N.T., Kowalczyk, R., Duval, B., van den Herik, J., Loiseau, S., Filipe, J. (eds.) Transactions on Computational Collective Intelligence XX. LNCS, vol. 9420, pp. 179–200. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27543-7_9
12. Genesereth, M., Love, N., Pell, B.: General game playing: overview of the AAAI competition. AI Mag. **26**(2), 62–72 (2005)
13. Kraus, S., Gan, R., Lehmann, D.: Designing and building a negotiating automated agent. Comput. Intell. **11**(972), 132–171 (1995)

14. Marinheiro, J., Lopes Cardoso, H.: A generic agent architecture for cooperative multi-agent games. In: Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART, vol. 1, pp. 107–118 (2017)
15. Nash, J.: Non-cooperative games. Ann. Math. **54**(2), 286–295 (1951)
16. Norman, D.: David's Diplomacy AI Page. http://www.ellought.demon.co.uk/dipai/. Accessed 16 May 2017
17. University of Essex. The GVG-AI Competition. http://www.gvgai.net. Accessed 16 May 2017
18. Reis, L.P., Mendes, P., Teófilo, L.F., Lopes Cardoso, H.: High-level language to build poker agents. In: Rocha, Á., Correia, A., Wilson, T., Stroetmann, K. (eds.) Advances in Information Systems and Technologies. AISC, vol. 206, pp. 643–654. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36981-0_59
19. Schreiber, S.: GGP.org - General Game Playing. http://www.ggp.org. Accessed 16 May 2017
20. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature **529**, 484–503 (2016)
21. Thielscher, M.: The general game playing description language is universal. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11), pp. 1107–1112 (2011)

Comparing the Effects of Disturbances in Self-adaptive Systems - A Generalised Approach for the Quantification of Robustness

Sven Tomforde[2(✉)], Jan Kantert[1], Christian Müller-Schloer[1],
Sebastian Bödelt[1], and Bernhard Sick[2]

[1] Institute of Systems Engineering, Leibniz University Hanover,
Appelstr. 4, 30167 Hanover, Germany
`{kantert,cms,boedelt}@sra.uni-hannover.de`
[2] Intelligent Embedded Systems Group, University of Kassel,
Wilhelmshher Allee 73, 34121 Kassel, Germany
`{stomforde,bsick}@uni-kassel.de`

Abstract. Self-adaptation and self-organisation (SASO) are increasingly used in information and communication technology to master complexity and keep the administrative effort at an acceptable level. However, using SASO mechanisms is not an end in itself – the primary goal is typically to allow for a higher autonomy of systems in order to react appropriately to disturbances and dynamics in the environmental conditions. We refer to this goal as achieving "robustness". During design-time, engineers have different possibilities to develop SASO mechanisms for an underlying control problem. When deciding which path to follow, an analysis of the inherent robustness of possible solutions is necessary. In this article, we present a novel quantification method for robustness that provides the basis to compare different control strategies in similar conditions.

1 Introduction and Motivation

Information and communication technology (ICT) has undergone a dramatic change: From single isolated fully comprehensible systems with a clear scope of functionality to large-scale structures of interconnected subsystems, see Bellman et al. (2014). Although the benefits of this change are clearly visible, this implies a severe risk: Complexity is everywhere, see Tomforde et al. (2014). And this complexity is what turns out to be a nightmare for developers, administrators, and even users. In order to be able to master complexity in ICT solutions, initiatives such as Organic Computing (OC), see Tomforde et al. (2017), and Autonomic Computing (AC), see Kephart and Chess (2003), emerged that proposed to mimic architectural and behavioural characteristics as found in natural and social systems, such as self-organisation, self-adaptation or decentralised control. As a result, traditional top-down organised and centralised design-concepts with single points of failure are replaced by bottom-up solutions that make use of cooperative efforts of distributed autonomous subsystems (or agents in terms of

© Springer International Publishing AG, part of Springer Nature 2018
N. T. Nguyen et al. (Eds.): TCCI XXVIII, LNCS 10780, pp. 193–220, 2018.
https://doi.org/10.1007/978-3-319-78301-7_9

Wooldridge (2009)) – with the goal to achieve desirable properties such as, e.g., self-healing, self-protection, and self-optimisation. We will refer to these systems as self-adaptive and self-organising (SASO) systems throughout this article.

Using SASO mechanism to control the behaviour of ICT systems is just the means and not the end: The primary goals are a reduction of administrative efforts and an increase of resilience against external or internal disturbances. The former goal is inherent in transferring control decisions to systems themselves, since the user or administrator is not bothered with decisions at lower abstraction levels. However, the latter goal is sometimes misconceived with the desire to allow for higher utility values. Conceptually, SASO systems are not per se faster than conventional systems but they return faster to a certain corridor of an acceptable utility in the presence of disturbances. The ultimate goal of SASO systems is, consequently, to become more resilient against disturbances and attacks[1] from outside. We refer to this property as "robustness" in this article.

In this article, we propose a novel technique to measure the property of robustness in SASO systems. This technique provides the basis for (i) allowing engineers to estimate the possible benefit of applying SASO concepts as a basis for a comparison against other solutions, (ii) allowing researchers in the domain of SASO systems to empirically evaluate their systems in a generalised manner, and (iii) providing a measurement foundation for testing SASO systems that takes inherent properties into account that differ from traditional systems engineering.

The general idea of the proposed technique to quantify robustness is to use the area of the characteristic utility degradation over time. Compared to other works from the state-of-the-art, this bears the advantage that this area captures the depth of the utility drop as well as the duration of the recovery. Intuitively, a degradation of zero corresponds to an ideally robust system. However, we require a more detailed analysis: The gradient of the drop phase and the bottom level of the degraded utility provide features that characterise the behaviour of the SASO mechanisms by means of a so-called *passive robustness*. As response to such a drop in utility, SASO mechanisms typically react with adaptations of the behaviour to guide the observed utility back to the favoured corridor of pre-specified target values. The upward gradient of this utility after starting the adaptation and recovery process is an indicator of the effectiveness of the systems active robustness. However, "utility" is an application-specific metric. It is used as a generalised term for the targeted effect of the system as defined by the user. Conceptually, it may be calculated by taking varying basic indicators into account, such as speed (in case of robots or autonomous cars), performance or throughput (in case of computing systems), or transfer rates (in case of communication systems).

This article is an extended and refined version of a conference paper, see Kantert et al. (2017), and is based on a preliminary concept as can be found

---

[1] An attack is a certain instance of the broader class of disturbances. In the remainder of this article, we will use the term "attack" but the discussion is valid in general for all kinds of disturbances.

in Edenhofer et al. (2016). Compared to the conference paper, we extended the formal approach to measure robustness, discussed the underlying system model in detail, and derived the limitations of the proposed solution in a formal manner.

The remainder of this article is organised as follows: Sect. 2 presents the system model for SASO systems as considered in this article and explains the generalised approach to quantify robustness in SASO systems. Afterwards, Sect. 3 compares this approach to work from the state-of-the-art. In order to investigate the general applicability and to highlight the benefits of the proposed technique, Sect. 4 applies it to a scenario from the field of wireless sensor networks (see Kantert et al. (2015)), Sect. 5 considers an example scenario from the field of open distributed systems for volunteered computing (see Kantert et al. (2014)), and Sect. 6 uses the Organic Traffic Control system – which is a SASO approach to urban traffic control and management, see Prothmann et al. (2011) – as example. The insights from these examples are discussed in Sect. 7 and the current limitations are explained in detail. Finally, Sect. 8 summarises the article and gives an outlook to future work.

2 Approach – Measuring Robustness

In the following section, we introduce our technique to measure the robustness of SASO systems as response to certain (external) disturbances or attacks. We initially present the underlying system model to provide a terminological basis of understanding for some of the key terms used in this article. Afterwards, we distinguish between passive and active robustness of the system and develop the quantification technique. This approach to classifying and quantifying the robustness is an extended version of our previous work which can be found in Kantert et al. (2016a) and Kantert et al. (2017).

2.1 System Model

From a conceptual point of view, we assume a self-adaptive and self-organising (SASO) system S to consist of a potentially large set of autonomous subsystems $a_i \in A$. Each a_i is equipped with sensors and actuators. Internally, each a_i distinguishes between a productive part (PS, responsible for the basic purpose of the system) and a control mechanism (CM, responsible for controlling the behaviour of PS and deciding about relations to other subsystems). This corresponds to the separation of concerns between *System under Observation and Control* (SuOC) and *Observer/Controller* tandem in the terminology of OC, see Tomforde et al. (2011). Figure 1 illustrates the basic system with its input and output relations. However, this is just used to highlight what we mean by referring to *autonomous* subsystems. In particular, the user guides the behaviour of a_i using U and does not intervene at decision level – actual decisions are taken by the CM.

The goal of this CM is to guide the behaviour of the PS: in response to environmental dynamics, internal status modifications, and external disturbances. To do so, we assume that a system utility is computable from environmental

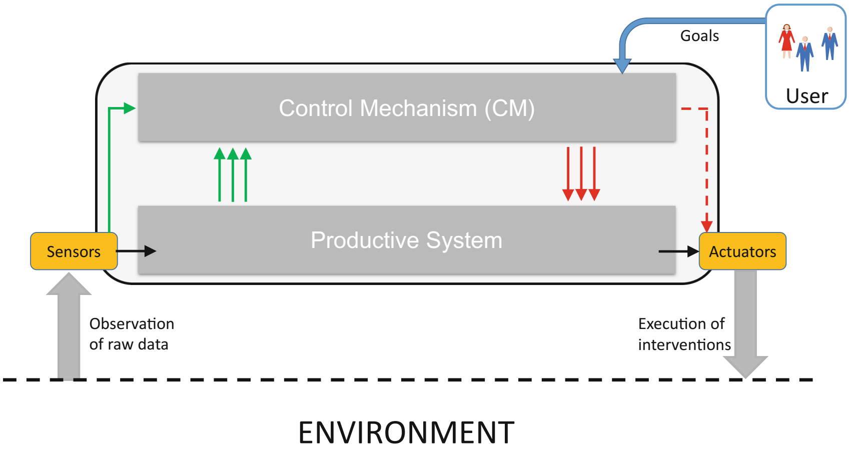


Fig. 1. System model of a SASO system. The behaviour of the productive system is guided by an internal control mechanism which acts on behalf of the user.

and internal data to estimate the success of the current control strategy. A reaction to disturbances is then considered as an attempt to maintain this utility and consequently counteract these disturbances.

2.2 Passive and Active Robustness

We assume a system S in an undisturbed state to show a certain target performance. More generally, we rate a system by a utility measure U, which can take the form of a performance or a throughput (in case of a computing system), a speed (in case of car), or any other application-specific metric. Typically, a system reacts to a disturbance by deviating from its target utility U_{target} by ΔU.

Passively robust systems, such as flexible posts or towers under wind pressure, react to the disturbance (i.e., the wind pressure) by a deflection $\Delta U = D_x$. This deflection remains constant as long as the disturbance remains. Active robustness mechanisms (such as self-adaptation effected by a CM realised as observer/controller tandem) counteract the deviation and guide the system back to the undisturbed state with $\Delta U = 0$ or U_{target}. If we want to quantify robustness (for comparison between different systems), we have to take into account the following observables:

1. The strength of the disturbance, z
2. the drop of the system utility from the acceptable utility U_{acc}, ΔU, and
3. the duration of the deviation (the recovery time $t_{\text{rec}} - t_z$).

We will introduce the developed method that takes all three aspects into consideration in the following part of this article.

2.3 Measuring Robustness

We assume that it is generally feasible to measure the utility over time (at least from the point of view of an external observer). However, it is often hard to quantify the strength of a disturbance z. Therefore, depending on the application, an estimation of z is required for our model. Furthermore, the system has to know a target utility value U_{target} (maybe the highest possible utility) and an acceptable utility value U_{acc} (a minimal value where the system is still useful to the user). This follows the ideas of a state space model for SASO systems as discussed in Schmeck et al. (2010). The goal of the model is to measure the response of a system to a certain disturbance in terms of its utility and compare it to other disturbances or systems.

In Fig. 2, we show a typical (artificial) utility function $U(t)$ as example. In the beginning, U is at the target value U_{target}. At time t_z, a disturbance of strength z happens and the utility (red) decreases. Once it drops below the acceptance threshold U_{acc} at t_{cm}, a control mechanism (CM) starts to intervene. At t_{low}, U reaches $U_{\text{low,perm}}$ without an effective recovery mechanism and $U_{\text{low,cm}}$ if a control mechanism is acting against the impact of the disturbance. With a CM, U starts to recover at t_{rec} and passes U_{acc} at t_{acc}. However, U does not recover without an appropriate CM realising the desired SASO capabilities.

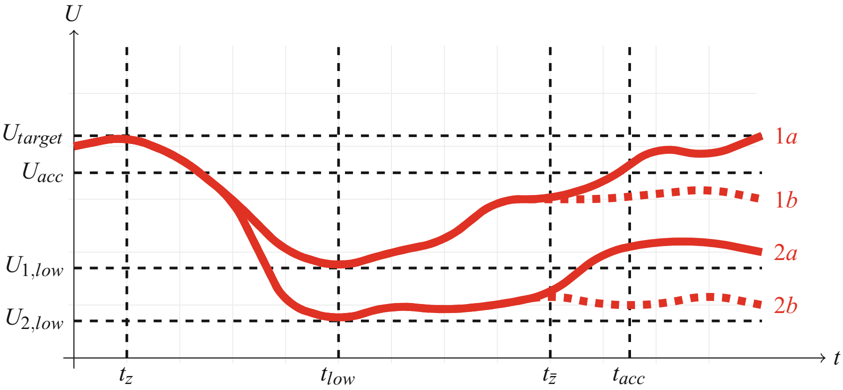


Fig. 2. Utility degradation over time. At t_z a disturbance with strength z occurs. The utility U drops. When it reaches U_{acc} (i.e. a utility value that is mapped on the acceptance space of the considered system, see Schmeck et al. (2010)), a control mechanism (CM) is activated for (1) which only decreases to $U_{\text{low,cm}}$ (i.e. the lowest utility value under presence of the control mechanism CM). At t_{rec}, recovery starts and (1) passes U_{acc} at t_{acc}. For (2) no CM is activated and utility drops to $U_{\text{low,perm}}$ (with *perm* referring to a system-inherent permanent robustness level) and no recovery occurs. When the attack ends (a), the utility recovers U_{target} eventually (i.e. it reaches the target space again, see Schmeck et al. (2010)). If the attack prevails or did permanent damage to the system (b), utility may stay at a lower value. (Color figure online)

We can differentiate between two classes of behaviour during an attack/disturbance:

(1) An effective CM is started and the system recovers to at least U_{acc}.
(2) The system does not recover during attack (or during a disturbance).

Furthermore, we see two different types of behaviour when the attack ends at $t_{\overline{z}}$. Either (a) the utility reaches the same value as before the attack (here U_{target} at t_{target}; solid red line in Fig. 2), or (b) it stays at the same level as during the attack (dashed line). In total, this results in four different stereotypes of behaviour:

(1a) The system S recovers during the attack to $U \geq U_{\text{acc}}$ and returns to $U \geq U_{\text{target}}$ when the attack ends. This is a *strongly robust system.*
(1b) S recovers during the attack to $U \geq U_{\text{acc}}$ and stays there when the attack ends. This is a *weakly robust system.*
(2a) S does not recover during the attack but returns to the previous value after the attack ends. Such a system shows just a certain “elasticity”, we call it *partially robust.*
(2b) S does not recover during the attack, it stays at the low utility level $U_{\text{low,perm}}$. Such a system is not robust at all.

The previous classification of system behaviour just gives an abstract overview of possible observations in response to disturbances and augments this with a verbal description of how we intend to consider the underlying robustness levels. However, to compare the effect of different disturbances z_j or different CMs, we have to quantify the utility degradation. We define the utility degradation D_U as the area between a baseline utility (U_{baseline}) and $U(t)$ for the time when $U \leq U$acc (see Eq. (1)):

$$D_U := \int_{t_z}^{t_{\overline{z}}} (U_{\text{baseline}}(t) - U(t))\, \mathrm{d}t \tag{1}$$

As shown in Fig. 3, we define a baseline U_{baseline} for the measurement. This can be either hypothetical by using U_{target} or U_{acc} (in Fig. 2) or we can run a reference experiment (as we will show later in Scenario 2, see Sect. 5). In order to maximise the system robustness, we have to minimise D_U. A system, which never drops below U_{baseline} apparently has maximal robustness. We define the robustness during attack R_a as:

$$R_a := \frac{\int_{t_z}^{t_{\overline{z}}} U(t)\, \mathrm{d}t}{\int_{t_z}^{t_{\overline{z}}} U_{\text{baseline}}(t)\, \mathrm{d}t}$$

Conceptually, we measure a utility degradation D_U for each possible CM under consideration. This allows for a comparison of the expected behaviour of different CMs during an attack (such as cases (1) and (2) from above) or the effectiveness of one CM for different attacks.

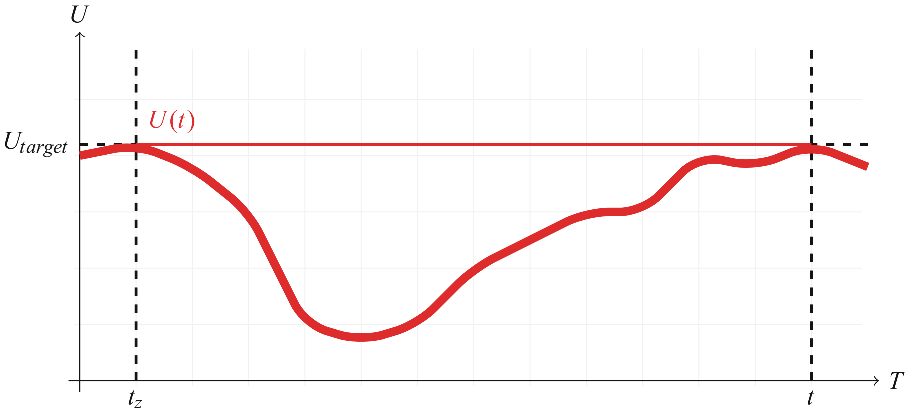


Fig. 3. Visualisation of D_U and D_{baseline}. The upper area is D_U and the relation between the two areas equals the robustness R.

In addition, we can measure the long-term utility degradation $D_{U,\text{long\_term}}$ when the attack ended – which is the area between U_{baseline} and the actual U for the time after $t_{\overline{z}}$. This allows us to compare the cases (1a/2a) and (1b/2b) from above. Hence, the long-term robustness R_l is defined as:

$$R_l := \frac{\int_{\overline{z}}^{t_{\text{target}}} U(t)\,\mathrm{d}t}{\int_{\overline{z}}^{t_{\text{target}}} U_{\text{baseline}}(t)\,\mathrm{d}t}$$

In cases where $U(t)$ never reaches U_{target} again, t_{target} is ∞ (also U_D is ∞) but we can calculate the open integral:

$$t_{\text{target}} = \infty \rightarrow R_l := \frac{\int_{\overline{z}}^{t_{\text{target}}} U(t)\,\mathrm{d}t}{\int_{\overline{z}}^{t_{\text{target}}} U_{\text{baseline}}(t)\,\mathrm{d}t} = \lim_{t\to\infty} \frac{U(\infty)}{U_{\text{baseline}}(\infty)}$$

In all “standard” cases, R is assumed to be in the interval $[0, 1]$. It can never be negative and will only be larger than 1 if $U(t)$ improves through the disturbance which happens only under very specific conditions (i.e., the system settled in a local optimum before the disturbance occurs, and is able to leave this local optimum as a result of the CM’s intervention). Even if the utility $U(t)$ never recovers, we get an asymptotic robustness value. However, values of R_l with $t_{\text{target}} = \infty$ are not comparable to values for R_l with $t_{\text{target}} \neq \infty$ because the open integral would be always 1 in this case.

2.4 Examples

In the following, we describe three exemplary scenarios which should be distinguishable using our approach. Furthermore, we demonstrate why our approach cannot be reduced to the individual components. In all cases, the dotted graph should be rated as more robust than the solid graph.

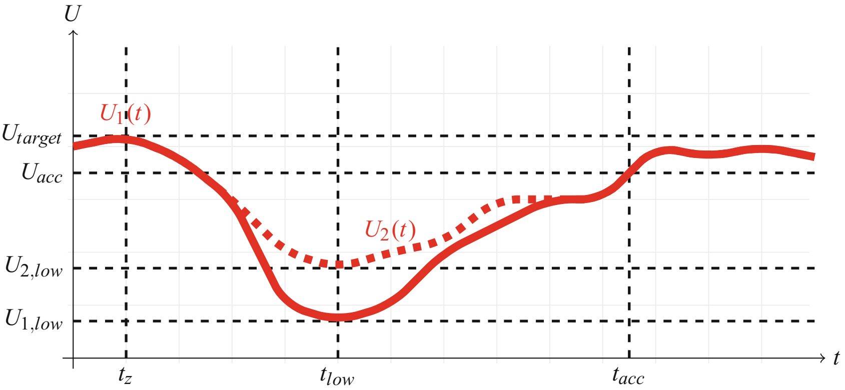


Fig. 4. Example 1. Utility over time for two experiments. The dotted graph drops less but t_{acc} is the same. Robustness for the dotted graph is better.

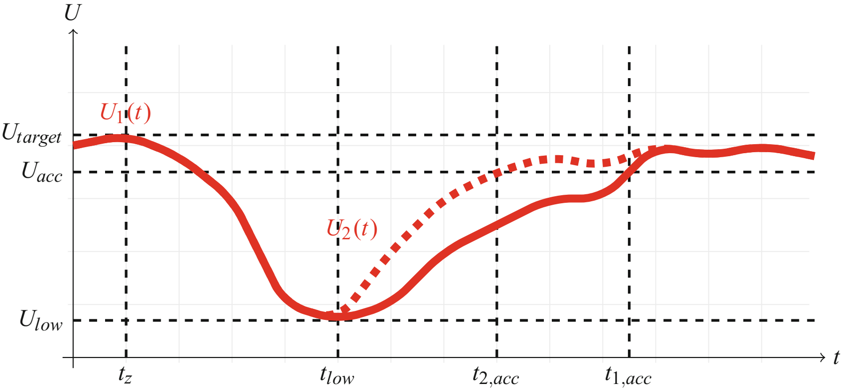


Fig. 5. Example 2. Utility over time for two experiments. Both graphs drop by the same amount but t_{acc} is smaller for the dotted graph. Robustness for the dotted graph is better.

Example 1 (see Fig. 4) shows the same graph as Fig. 3 which recovers at the end of the experiment. The dotted graph drops slightly less than the solid one. However, t_{acc} has the same value for both graphs which shows that the duration alone is clearly not sufficient to quantify the robustness because the dotted graph does not drop as much. Nevertheless, the integral of the utility degradation D_U is obviously smaller and the robustness is better for the dotted graph.

Furthermore, Example 2 (see Fig. 5) highlights a case where both graphs drop similarly. However, the dotted graph recovers faster and t_{acc} is smaller which demonstrates that the drop alone is also not sufficient. Again, our approach using an integral will prefer the dotted graph over the solid one since it results in the higher overall robustness value.

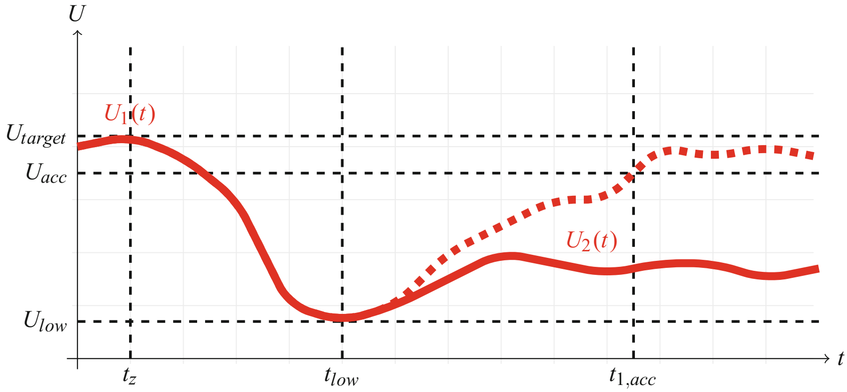


Fig. 6. Example 3. Utility over time for two experiments. Both graphs drop by the same amount but only the dotted graph recovers. Robustness for the dotted graph is better.

In Example 3 (see Fig. 6), the solid graph does not recover within the experiment (e.g. t_{acc} is inf). However, the dotted graph recovers and both graphs drop similarly. In this case, the robustness of the solid graph is not clearly better when only considering the results of computing the integrals. Nevertheless, this marks one of the limits of our approach and we further discuss this in Sect. 7.2.

3 Related Work

The term "robustness" is widely used with different meanings in literature, mostly depending on the particular context or underlying research initiative. Typical definitions include the ability of a system to maintain its functionality even in the presence of changes in their internal structure or external environment (sometimes also called *resilient* or *dependable* systems), see Callaway et al. (2000), or the degree to which a system is insensitive to effects that have not been explicitly considered in the design, see Slotine et al. (1991).

When especially considering engineering of solutions driven by information and communication technology, the term "robust" typically refers to a basic design concept that allows the system to function correctly (or, at the very minimum, not failing completely) under a large range of conditions or disturbances. This also includes dealing with manufacturing tolerances. Due to this wide scope of related work, the corresponding literature is immense, which is e.g. expressed by detailed reports that range back to the 90ies, see e.g. Taguchi (1993). For instance, in the context of scheduling systems, robustness of a schedule refers to its capability to be executable – and leading to satisfying results despite changes in the environment condition, see Scholl et al. (2000). In contrast, the systems we are interested in (i.e., SASO systems) typically define robustness in terms of fault tolerance, see e.g. Jalote (1994).

In computer networks, robustness is often used as a concept to describe how well a set of service level agreements are satisfied. For instance, Menascé et al. explicitly investigate the robustness of a web server controller in terms of workloads that exhibit some sort of high variability in their intensity and/or service demands at the different resources, see Menascé et al. (2005).

Beyond these general definition, a quantification of robustness is part of a framework for measuring SASO behaviour. Existing metrics to identify SASO properties are either highly domain specific and hard to generalise or restricted to simple, small-scale models, see Parunak and Brueckner (2001), Chan (2011). Only limited research effort has been spent on metrics to determine the effort and the benefit of adaptation in distributed collections of autonomous subsystems, see Eberhardinger et al. (2015) for an overview. Examples include the relation between working and adaptation time, the availability of subsystems for task processing, and the performance of the overall system (i.e., the degree to which a certain goal is achieved). Kaddoum et al. (2010) discuss the need to refine classical performance metrics to SASO purposes and present specific metrics for self-adaptive systems. They distinguish between nominal and "self-*" situations and their relations. For instance, they measure the operation time in relation to the adaptation time to determine the effort. Cámara et al. (2014) investigated some of the developed metrics in detail for software architecture scenarios. In Schwind et al. (2013), the authors modelled the behaviour of a technical system as trajectory traversing a state space – and conceptually approached the resilience in terms of a dynamic constraint-based agent model. In particular, they proposed to capture the notion of resilience for dynamic systems using factors such as resistance, recoverability, functionality, and stability. However, a concrete definition for SASO systems is still open. In addition, success and adaptation efforts and ways to measure autonomy have been investigated, see e.g. Gronau (2016).

Only a few approaches known in literature aim at a generalised method to specifically quantify robustness. In a majority of cases, self-organised systems are either shown to perform better (i.e. achieve a better system-inherent utility function) or react better in specific cases (or in the presence of certain disturbances), see e.g. ICAC (2015) and SASO (2015). In the following, we discuss the most important approaches to robustness quantification.

In the context of Organic Computing (OC), a first concept for a classification method has been presented in Schmeck et al. (2010). Here, the idea is (as in our approach) to take the system utility into account. Based on a pre-defined separation of different classes of goal achievement (i.e. distinguishing between target, acceptance, survival, and dead spaces in a state space model of the system S), the corresponding states are assigned to different degrees of robustness. Consequently, different systems are either strongly robust (i.e., not leaving the target space), robust (i.e., not leaving the acceptance space), or weakly robust (i.e., returning from the survival space in a defined period of time). In contrast to our method, a quantitative comparison is not possible. In particular, this robustness classification does not take the recovery time into account.

Closely related is the approach presented in Nafz et al. (2011), where the internal self-adaptation mechanism of each element in a superior self-organising system has the goal to keep the element's behaviour within a pre-defined corridor of acceptable states. Using this formal idea, robustness can be estimated by the resulting goal violations at runtime. Given that system elements have to obey the same corridors, this would also result in a comparable metric. Recently, the underlying concept has been taken up again to develop a generalised approach for testing self-organised systems, where the behaviour of the system under test has to be expressable quantitatively, see Eberhardinger et al. (2015). However, this depends on the underlying state variables and invariants that are considered – which might be more difficult to assess at application level (compared to considering the utility function in our approach).

In contrast, Holzer et al. considered subsystems in a system S as nodes and models them as stochastic automaton in a network. In addition, the nodes' configuration is modelled as a random variable, see Holzer and de Meer (2011). Based on this approach, the authors propose to compute the level of resilience (the term is used there similarly to robustness in this article) depending on the network's correct functioning in the presence of malfunctioning nodes that are again modelled as stochastic automatons, see Holzer and de Meer (2009). In contrast to our approach, this does not result in a comparable metric and limits the scope of applicability due to the underlying modelling technique.

From a multi-agent perspective, Di Marzo Serugendo approached a quantification of robustness using the accessible system properties, see Di Marzo Serugendo (2009). In general, properties are assumed to consists of invariants, robustness attributes, and dependability attributes. By counting or estimating the configuration variability of the robustness attributes, systems can be compared with respect to robustness. Albeit the authors discuss an interesting general idea, a detailed metric is still open.

Also in the context of multi-agent systems, Nimis and Lockemann (2004) presented an approach based on transactions. They model a multi-agent system as a layered architecture (i.e. seven layers from communication at the bottom to the user at the top layer). Of particular interest is the third layer, i.e. the conversation layer. The key idea of their approach is to treat agent conversations as distributed transactions. The system is then assumed to be robust, if guarantees regarding these transactions can be formulated. This requires technical prerequisites, i.e. the states of the participating agents and the environment must be stored in a database – this serves as a basis for formulating the guarantees. Obviously, such a concept assumes hard requirements that are seldom available, especially not in open, distributed systems where system elements are not under control of a centralised element and might behave unpredictably.

To conclude this discussion, we can observe that a generalised approach to quantify robustness is needed that: (a) works on externally measurable values, (b) does not need additional information sources (e.g. transactional data-bases), (c) distinguishes between system-inherent (or passive) and system-added (or active) robustness (to allow for an estimation of the effectiveness of the particular mechanism) and (d) comes up with a measure that allows for a comparison of different systems for the same problem instance. We claim that our approach as outlined before fulfills all of these aspects. In order to demonstrate the effectiveness, we apply it to three different use cases in the following part of this article.

4 Application Scenario 1: Wireless Sensor Networks

As a first example to analyse the behaviour of the presented measurement technique, we consider Wireless Sensor Networks (WSNs). Such a WSN typically consists of spatially distributed nodes which communicate over a radio interface. Nodes sense locally and send the result to the root node in the network. Since most nodes cannot reach root directly, other nodes have to relay packets. To find a path to root, the *Routing Protocol for Lossy and Low Power Networks* (RPL) is used, see Winter et al. (2012). The primary objective (O1) in such networks is to reach a high Packet Delivery Rate (PDR; ranges from 0 to 1). Since nodes are battery-powered, the secondary objective (O2) is to minimise the number of Transmitted Packets (#TX) because sending data over the air causes most power consumption in WSNs. These two objectives translate into two utility functions, which we have to investigate with respect to their robustness. Utility function 1 is PDR(t), utility function 2 is #TX(t).

In open distributed sensor networks, attacks by malicious or broken nodes can occur, which lead to poor PDR. To counter such threats, we introduced end-to-end trust in RPL in a previous work, see Kantert et al. (2016b). In this trust-enhanced approach, the nodes assess the trustworthiness of their parents and isolate bad-behaving nodes. This constitutes a specific self-organised control mechanism CM. We are interested in a comparison of different variants of CMs:

- CM0 OF0. This is the default routing mechanism in RPL as described in Winter et al. (2012). It selects parents by the smallest rank.
- CM1 Trust + ETX. Nodes use a trust metric to rate and isolate bad-performing parents. Once a node is isolated, it is not used for communication paths anymore. Kantert et al. (2016b) presents the approach in detail. The particular method is not of interest in the context of this article, it serves as a representative for a more powerful, self-organised control mechanism.
- CM2 Trust + ETX + Second Chance. This approach is similar to the previous one but also incorporates a mechanism to retry previously isolated parents occasionally (i.e. after re-stabilising the system, see again Kantert et al. (2016b) for details). This approach serves as a representative with even more decision freedom of the CM.

In an undisturbed RPL network, our trust-enhanced system implemented as CM1 behaves very similar to standard RPL (CM0). However, when an attack occurs, standard RPL looses numerous packets and PDR drops because it cannot handle (intentional or unintentional) malicious behaviour. When enabling our

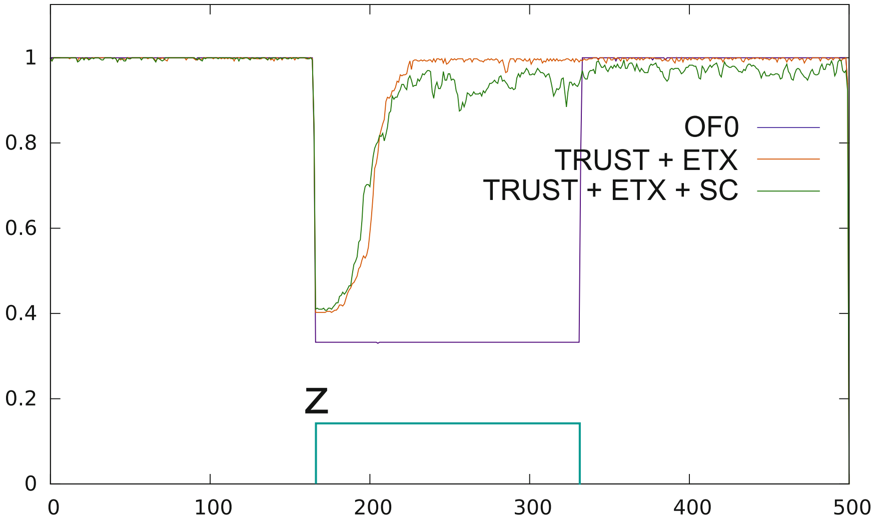


Fig. 7. Objective 1, Utility function 1: Packet Delivery Rate (PDR) over sequences (time). At time step 160, an attack starts and ends at time step 340. PDR drops to about 30% for OF0 during the attack and recovers afterwards. TRUST + ETX and TRUST + ETX + Second-Chance recover during the attack and stay near 100% PDR until the end of the experiment, see Kantert et al. (2017).

approach, nodes start to identify and isolate bad-behaving parents. Hence, the PDR recovers to nearly 100% (i.e. U_{target}) while the attack happens. Standard RPL only recovers after the attack ends (see Fig. 7).

For O1 (PDR(t)), D_U is 112 for CM0, 13.7 for CM1 and 17.9 for CM2. Quantitatively, D_U of CM1 is 87.7% better than CM0, and CM2 is 84% better than CM0. Also, CM1 has 23.5% smaller D_U than CM2. The baseline is 1 (see Eq. (2)). R_a for CM0 has a value of 70% (3), CM1 has a value of 91% (4) and CM2 has a value of 89% (5).

$$U_{\text{baseline}}(t) := 1 \tag{2}$$

$$R_{a,\text{CM0}} \approx 70\% \tag{3}$$

$$R_{a,\text{CM1}} := \frac{146.3}{160} \approx 91\% \tag{4}$$

$$R_{a,\text{CM2}} := \frac{142.1}{160} \approx 89\% \tag{5}$$

For objective O1 (i.e. PDR(t)) CM1 recovers perfectly during the attack and returns to the same value as before the attack. Therefore, in this scenario, RPL with Trust + ETX is rated as strongly robust. However, this changes when we look at the second objective O2 represented by utility function 2: metric #TX(t) (the number of transmitted packets, see Fig. 8). When the attack starts, #TX drops for all CMs (which would be perfect if PDR stayed at a constant level). For Trust + ETX it increases to a higher level because the new routes are longer and

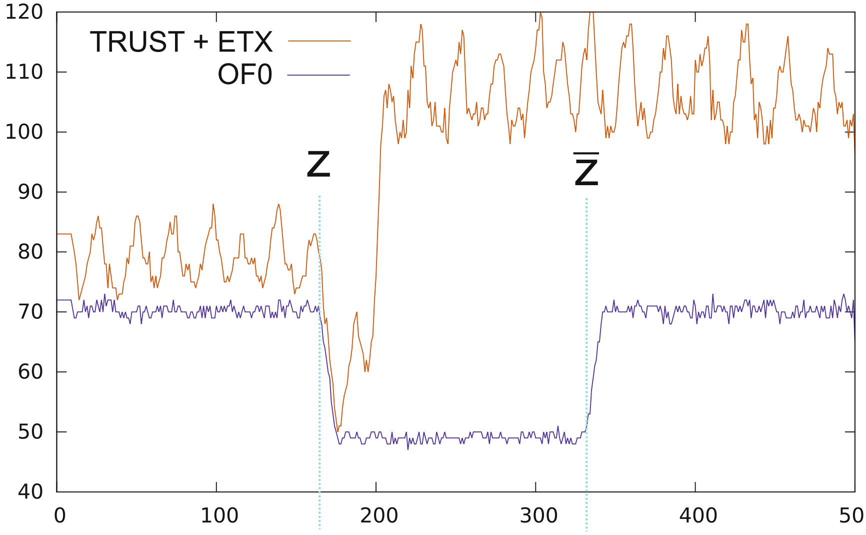


Fig. 8. Objective 2, Utility function 2: Transmitted Packets (#TX) over time. At time step 160, an attack starts – which ends at time step 340. During the attack #TX increases for the TRUST + ETX control mechanism (CM1) and stays at that level after the attack. For OF0 (CM0), #TX drops during the attack and recovers to the undisturbed level afterwards, see Kantert et al. (2017).

require more transmissions. This is expected since some parents failed. However, after the attack ends, only standard RPL returns to its previous #TX. Trust + ETX stays at a high level. This is acceptable when dealing with intentional malicious attackers but this is bad when disturbances are only temporary. Therefore, Trust + ETX is not robust regarding the second utility function.

Most energy in wireless sensor nodes is used when sending packets. Thus, to recover the energy consumption and #TX after an attack, we introduced TRUST + ETX + Second-Chance as a third control mechanism CM2 which retries parents when the system has stabilised (from a node's local perspective). This leads to a slight decrease of the PDR during the attack because nodes loose packets when retrying during an attack. However, after the attack the PDR is very similar and #TX recovers to its level from before the attack (see Figs. 7 and 9).

For O2, we measure D_U only after the attack. Quantitatively, D_U for CM1 is ∞ because it does not return to the previous values. For CM0, D_U is about 0. CM2 needs some time to recover and has a D_U of 1998.6. Thus, CM0 is the best metric when only considering O2 (since it is very bad for O1). In our robustness metric, we have to assume that a higher utility is better. In this case, the best utility for O2 would be a value of 0 (which is unrealistic). Therefore, we invert $U(t)$ to get a utility function for the calculation of R_l (see Eq. (6)). This results in a baseline U_{baseline} of 43 (saved packets per second; see Eq. (7)). For CM1 this results in an asymptotic long-term robustness R_l of 35% (8). CM2 recovers and

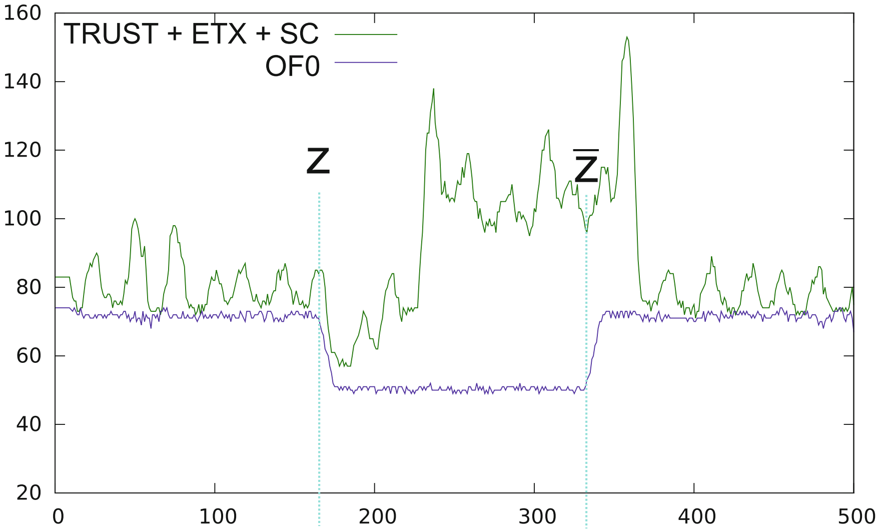


Fig. 9. Objective 2, Utility function 2: Transmitted Packets (#TX) over time. At time step 160, an attack starts, and ends at time step 340. During the attack #TX(t) increases for the TRUST + ETX + Second-Chance (CM2). For OF0 (CM0), #TX(t) drops during the attack and recovers to the same level as before afterwards. Unlike in Fig. 8, TRUST + ETX + Second-Chance (CM2) recovers to a similar level as before the attack, see Kantert et al. (2017).

has a robustness value R_l of 47% (not comparable to CM1; Eq. (9)).

$$U_{\text{normalised}}(t) := 120 - U(t) \tag{6}$$

$$U_{\text{baseline}} := 120 - 77 = 43 \frac{\text{packets}}{s} \tag{7}$$

$$R_{l,\text{CM1}} \approx \frac{120 - 105}{43} \approx 35\% \tag{8}$$

$$R_{l,\text{CM2}} \approx \frac{401.4}{860} \approx 47\% \tag{9}$$

CM2 (TRUST + ETX + Second-Chance) is strongly robust for PDR. Also, it is robust regarding #TX. It does not recover during the attack (because better routes do not exist) but it recovers to the previous state when the attack ends.

5 Application Scenario 2: Open Distributed Systems

Another scenario we measured robustness in is the simulation of an open, distributed Multi-agent System with applied trust metrics, the *Trusted Desktop Grid* (TDG) Edenhofer et al. (2015). In this grid, jobs J are created by the agents and split into work units (WU), which are calculated in a distributed way by the agents. The success (or utility) is expressed as the speedup $\sigma(t)$. $\sigma(t)$ is defined as the time the agent would have needed to process all work units on its own ($self$) divided by the real time it took to calculate the job in a distributed way in the system ($dist$), see Eq. 10. A job J is released in time step t_J^{rel} and completed in t_J^{compl}.

$$\sigma = \frac{\sum\limits_J (t_{self}^{compl} - t_{self}^{rel})}{\sum\limits_J (t_{dist}^{compl} - t_{dist}^{rel})} \tag{10}$$

In this scenario, our objective is to maximise the utility function speedup $\sigma(t)$ (see Eq. (10)). Nodes make decisions based on a local trust metric and isolate bad-behaving agents using this Control Mechanism (CM1; see Klejnowski (2014) for details). We compare different attacks:

A0 No attack. Used as baseline.
A1 A short attack which ends at $t_{\overline{z}}$.
A2 Permanent attack which continues until the experiment ends.

Additionally, we consider two different disturbances by stereo-type attacker behaviour:

D1 *Egoistic agents* (EGO) are accepting all WU, but abort 80% of them after some time, which decreases $\sigma(t)$ because these WUs have to be redistributed.
D2 *Freeriding agents* (FRE) do not accept WU at all. They reject WUs right away but try to distribute their WUs at the same time.

For evaluation purposes, we investigated exemplary scenarios that incorporate disturbed system states. More precisely, we simulated 100 well-behaving adaptive agents (ADA). An ADA is called "adaptive" since it modifies its interaction strategy with another agent in response to own experiences and the system-wide reputation. In particular, this means that own bad experiences and/or bad experiences of others with this particular possible interaction partner will decrease the probability of interaction, see Edenhofer et al. (2016) for details.

At time t_z, 100 bad-behaving egoistic agents (EGO; D1) join the system, simulating a colluding attack. To show the effect of robustness in the system, we calculated the average speedup of 10 runs with a length of 200,000 ticks each. We want to compare the system behaviour (i.e. its robustness) under two different attacks. Attack A1 starts at tick 80,000 and lasts 40,000 ticks. Attack A2 is a continuous attack of 100 EGO starting at tick 80,000. As baseline, we ran the same experiment without an attack (A1; see Fig. 10).

In both attacks (A1 and A2), the average speedup of the ADA during the attack of the EGO is decreased, due to the malevolent behaviour of the EGO. t_{low} marks the point in time, where σ of the ADA is at its lowest. During the attack, once the average speedup of ADA increases by 5% (t_{rec}) over $\sigma(t)$ at t_{low}, the recovery phase is said to start; recovery is defined to be reached, if $\sigma(t)$ is at least 75% (at t_{acc}) of $\sigma(t)$ before the attack in t_z.

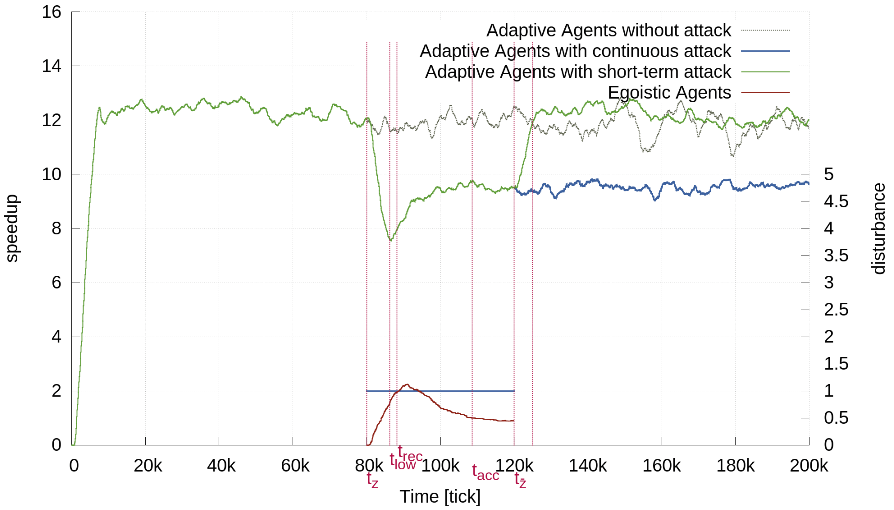


Fig. 10. Disturbance D1: Robustness in the TDG. Attack of 100 EGO from tick 80,000 to 120,000 and continuous from 80,0000 (A2). t_{low} marks the point in time, where $\sigma(t)$ is at its lowest. The speedup starts to recover (t_{rec}) during the attack until it reaches U_{acc} (at t_{acc}). After the attack has stopped, the speedup returns to the level it had before for A1. However, it stays at about 10 for A2.

In the first attack, after tick 120,000, the ADA have to redistribute the WUs formerly occupied by EGO. After these WUs have been successfully calculated, $\sigma(t)$ recovers to the level it had before the attack. In both attacks, $\sigma(t)$ of the ADA recovers during the attack until it reaches the acceptance space (i.e. system states where the utility is above U_{acc}). This is due to the EGOs getting low trust ratings because they abort their assigned WUs. Yet, they manage to retain a high enough reputation to still get some WUs computed by other agents. As 80% of the WUs held by EGOs have to be redistributed, $\sigma(t)$ of the ADA cannot recover to the level it had before the attack.

Quantitatively, D_U is 130,105 below the baseline for both A1 and A2. After the attack, D_U is 6250 for A2. D1 has a permanent influence and CM1 is not able to fully mitigate the effect. We measured the baseline utility U_{baseline} in a reference experiment. R_a for A1 and A2 is with 72% (see Eq. (11)). Similar to O2 in Scenario 1, only A2 fully recovers. For A1, we calculate R_l in an open integral in Eq. (12). Again, not comparable, R_l for A2 can be calculated in a closed integral (see Eq. (13); the open integral would have a value of 1).

$$R_{a,\mathrm{A1}} = R_{a,\mathrm{A2}} := \frac{349,895}{480,000} \approx 73\% \tag{11}$$

$$R_{l,\mathrm{A1}} \approx \frac{9.5}{12} \approx 79\% \tag{12}$$

$$R_{l,\mathrm{A2}} := \frac{113,750}{120,000} \approx 95\% \tag{13}$$

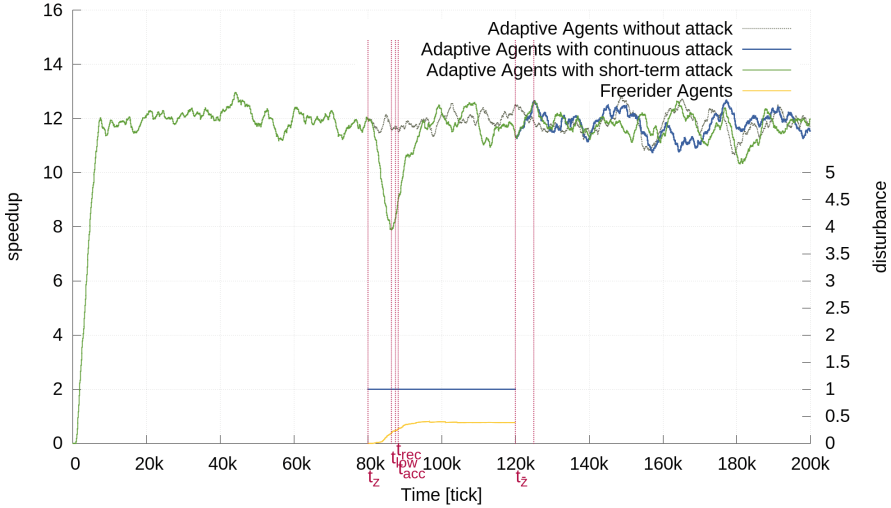


Fig. 11. Disturbance D2: Robustness in the TDG. Attack of 100 FRE from tick 80,000 to 120,000 (A1) and continuous from 80,0000 (A2) t_{low} marks the point in time, where σ(t) is at its lowest. The speedup starts to recover (t_{rec}) during the attack and quickly reaches U_{acc} and U_{target} for both A1 and A2. At tick 120,000 A1 and A2 are the same because the attackers are fully isolated and have no further influence Kantert et al. (2017).

Similarly, we run experiments for disturbance D2 with free-riding agents (see Fig. 11). Since those agents are simpler to detect, they are isolated within the attack period of A1 and the system fully recovers to U_{target} within the attack. U_D is $13,056$ for both A1 and A2 during the attack. After the attack ends, A1 is already fully recovered and the utility for A1 and A2 is similar. CM1 is able to fully mitigate this disturbance D2. R_a is 97% (see Eq. (14)). Since $U(t)$ fully recovers for both attacks, R_l is about 100% for A1 and A2.

$$R_{a,\text{D2}} := \frac{466,944}{480,000} \approx 97\% \tag{14}$$

6 Application Scenario 3: Urban Traffic Management

A third scenario that is often used as a basis to investigate SASO mechanisms in the context of a specific application domain is traffic control and management in urban regions. The reason for choosing this example application lies in its inherent dynamics and distributed nature, see Bazzan and Klügl (2009); Dinopoulou et al. (2006) for instance. One of the major SASO-based contributions in this context is the *Organic Traffic Control* (OTC) system, see Prothmann et al. (2011), that applies the observer/controller approach as well as an autonomous and safety-oriented learning process to the traffic domain.

In OTC, each intersection of the inner-city road network is managed by an observer/controller (following the general concept as defined in Tomforde et al. (2011)) instance that gathers detector data about the underlying traffic conditions (in terms of flows passing each turning movement) and reacts by adapting green durations. The success of the control strategy is typically expressed as flow-weighted delays:

$$t_D = \frac{\sum_i (M_i \times t_{d,i})}{\sum_i M_i} \tag{15}$$

In this formula, M_i corresponds to the current traffic flow at the i-th turning of the observed intersection and $t_{d,i}$ denotes the average waiting time with respect to a single turning t_i. This metric is also referred to as *Level of Service* (LoS), see Transportation Research Board (2000). In addition, the same metric is used to determine the most promising progressive signal system – which is described in detail in Tomforde et al. (2010).

Adapting traffic signalisation to changing demands and coordinating intersection controllers provides a first step towards robustness – but this does not dissolve the initial disturbance. In terms of the previously developed notion, this accounts as passive robustness. Re-routing of traffic participants counters the disturbance (i.e. a blocked road or traffic jams) directly and can be considered as active robustness mechanism in this context. In previous work, we equipped the Organic Traffic Control system (OTC) with such a mechanism Prothmann et al. (2012). Based on ideas resembling the Link State or Distance Vector Routing protocols as known from the computer networks domain, see Tanenbaum (2002) for instance, information about the shortest routes are exchanged between intersection controllers and provided to drivers at each incoming intersection, see Prothmann et al. (2012; 2011) for details.

In this scenario, the objective is to maximise the utility function *traffic flow* which can be measured globally. At t_z, one road is blocked until $t_{\overline{z}}$ (attack A1, red road in Fig. 12). We compare three different control mechanisms:

CM0 No active control mechanism. Used as a baseline.
CM1 OTC without routing.
CM2 OTC with routing.

The evaluation has been conducted for a simulated network that is illustrated in Fig. 12. The network consists of three Manhattan-type sub-networks. It contains 27 signaled intersections (depicted as circles) and 28 prominent destinations (depicted as diamonds). Within each sub-network, the intersections are connected by one-laned road segments of 250 m length that provide two additional turning lanes starting 100 m before an intersection. Regions are connected by two-laned roads. Signalised intersections are operated by an observer/controller (i.e. OTC) and can provide route recommendations for the prominent destinations. Each destination also serves as origin for traffic entering the network.

We configured the simulation as follows: eight vehicles per hour travel from every origin to every destination. Since this demand does not cause significant jams at the network's intersections, the scenario allows evaluating the impact of routing mechanisms under uncongested conditions. As disturbance, we temporarily blocked one of the roads connecting two sub-networks due to an incident (see mark in Fig. 12). The blockage affects both directions of the road, occurs after 25 min and lasts for 40 min within the simulation period. The incident scenario allows analysing the impact of the two different adaptation mechanisms (CM1 = OTC without routing, and CM2 = OTC with routing) in comparison to a standard fixed-time control approach (=CM0).

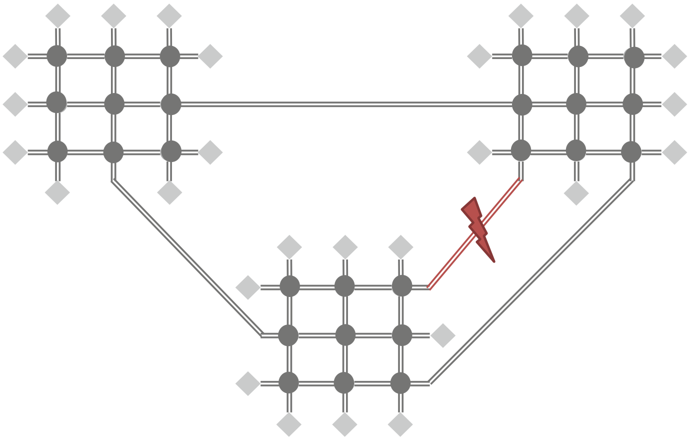


Fig. 12. An exemplary model of 27 intersections (depicted as grey circles), forming three connected Manhattan-style road networks. Traffic enters and leaves the network at locations referred to as destinations (depicted as light-grey diamonds). In the evaluation, the red road is blocked during attack A1, cf. Kantert et al. (2017). (Color figure online)

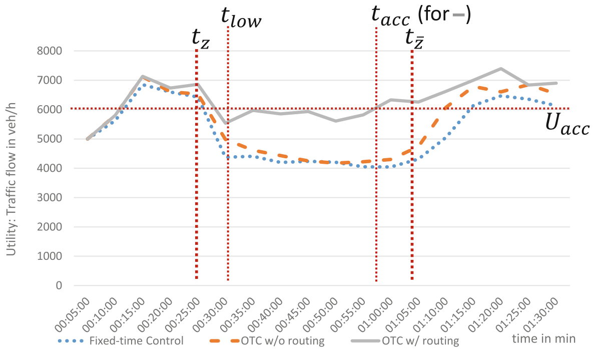


Fig. 13. Utility over time for OTC. At t_z one road is blocked (attack A1) and traffic has to be redirected. With fixed-time control (CM0), utility drops by nearly 3,000 vehicles per hour. OTC without routing (CM1) performs slightly better. Compared, OTC with routing (CM2) only drops about 1,000 vehicles per hour. After the attack A1 ends, all CMs recover the system to their previous utility Kantert et al. (2017).

Within the professional simulator Aimsun, cf. Barceló et al. (2005), we performed five runs of experiments for each instance under evaluation and computed averages. Figure 13 depicts the achieved results. We can clearly observe that the typical system behaviour of a disturbed and recovering system is visible in this scenario. When considering the OTC system without routing in comparison to a standard fixed-time control, we can distinguish two effects: (a) the fixed-time controller (i.e. CM0) defines the base-line (i.e. the maximum loss of utility) and (b) the standard OTC system (CM1) already provides a permanent robustness increase that improves the behaviour even if no disturbance takes place. If activating the routing mechanism (CM2), an active robustness is added. Quantitatively, U_D for CM1 is about 5.4% better than the reference solution (CM0) (i.e. 5534 vs 5250 vehicles per hour) and routing (CM2) adds further utility, i.e. an improvement in U_D of 14.0% (i.e. 6312 vehicles per hour).

$$U_{\text{baseline}}(t) := 7,000 \frac{\text{vehices}}{h} = 116.7 \frac{\text{vehicles}}{\text{min}} \tag{16}$$

$$R_{a,\text{CM0}} := \frac{87.5}{116.7} \approx 75\% \tag{17}$$

$$R_{a,\text{CM1}} := \frac{92.2}{116.7} \approx 79\% \tag{18}$$

$$R_{a,\text{CM2}} := \frac{105.2}{116.7} \approx 90\% \tag{19}$$

To calculate the robustness, we use a baseline based on U_{target} (see Eq. 16). Using the U_D values from above, we calculate the robustness R_a for all CMs in Eqs. (17), (18) and (19). What we can see in the figure is that a passive robustness is already in place with CM0, since most of the network is still operating without strong impact of the disturbance (in terms of traffic flow through the network). This changes if we just consider the blocked link: Here, for all three mechanisms the utility drops to zero and recovers immediately when the blockade is removed (not shown in figure). Hence, the robustness is achieved at network-level, since participants are routed using the best available link (i.e. with routing mechanism, CM2) or at least the non-affected participants benefit due to longer green durations (i.e. a blocked road receives a lower fraction of the phase cycle time when using OTC, CM1 and CM2). This mechanism may serve as a basis to compare the success of the routing mechanisms to other solutions from the state-of-the-art (such as Wedde et al. (2007) for self-organised routing or Dinopoulou et al. (2006) for autonomous traffic controllers) in terms of quantifying the varying robustness levels in different disturbance scenarios.

7 Discussion and Limitations

After applying the developed measurement technique to three example use cases from the wireless sensor network, trusted grid computing, and urban traffic management domains, we discuss the derived (generalised) insights in the following section. This discussion is augmented with an explanation of the current limitations of the proposed method which are addressed in current and future work.

7.1 Discussion

We demonstrated that our approach is applicable to three varying scenarios from different application domains. In Scenario 1, we compared three different control mechanisms with two different objective functions for wireless sensor networks. The resulting values are reasonable and allow for a simple and effective comparison of robustness levels for the considered CMs. In Scenario 2, we compared two different attack types and two different disturbances in a grid scenario. We measured the long-term influence for the disturbances and found that only one of the disturbances has a permanent effect. Afterwards, in Scenario 3, we measured the robustness for three CMs in a traffic management scenario.

The general shape of the utility graph was very similar in all our scenarios. Still, the utility degradation U_D is not comparable between scenarios because it has an application specific unit (i.e., it is hard to compare apples and oranges). However, our robustness metrics R_a and R_l are unit free and always have the same boundaries. Thus, they allow for some (abstract) comparison between different systems.

Nevertheless, some limitations apply: The approach requires to consider a fixed attack length $t_{\overline{z}} - t_z$. This limits the applicability of the concept to those cases where a disturbance is observable in the first place. Current research addresses this topic by developing techniques to deal with this issue. Also, it influences the value of R as illustrated by Scenario 2. If we had set $t_{\overline{z}}$ earlier the robustness R_a for both CMs would be lower. A similar challenge occurs when calculating R_l for two CMs which reach U_{target} at different times. To make their robustness R_l values comparable, a later t_{target} has to be chosen. This issue can be solved by observing the utility behaviour over time and adjusting the key values correspondingly – but it requires some caution when designing experiments.

We noticed another challenge when calculating R_l for O2 in Scenario 1 because our goal was to minimise the metric. However, for a utility function $U(t)$ a higher value should be better. Hence, we had to invert the objective function to calculate the utility. There is no need to normalise the utility function but larger values must be better which is important to keep the robustness values inside the interval between 0 and 1.

In addition, we saw in Scenario 1 that it is crucial to choose the right utility function. If we only consider O1, A1 has the best robustness. However, if we also take into account O2, A2 is clearly better. The remaining challenge here is to properly combine multiple metrics.

Finally, a more general remark is that we need a threshold defining an (at least) acceptable utility level. This may not be available in all cases. However, reinforcement learning concepts may be used to learn a situation-to-achievable-utility mapping, i.e. to estimate the best achievable utility for each occurring condition. This may then be combined with either a static interval defining acceptable behaviour or a dynamic boundary for utility drop that considers statistical properties such as variance to define an acceptable level of utility as a basis for calculating the robustness.

7.2 Limitations

We kept our approach as simple as possible and reduced the robustness measurement to a single value. Unfortunately, this induces certain limitations (as already mentioned at the end of Sect. 2.4). Especially cases where the utility does not recover post a challenge. Since we have to calculate the robustness using open integrals, only the last utility value matters. Figure 14 illustrates such a case: The solid and the dotted line both result in exactly the same robustness value because they converge to the same utility value at the end. However, the dotted graph is certainly better because it drops less.

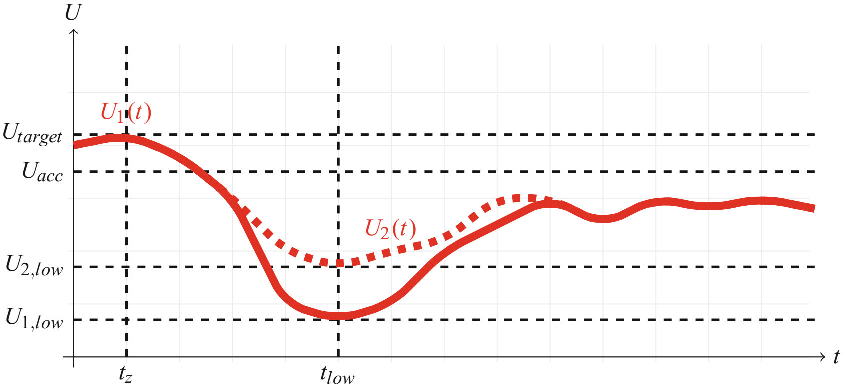


Fig. 14. Utility over time for two experiments. The dotted graph drops less. Both experiments do not recover. Robustness for the dotted graph should be better. However, our approach calculates the same robustness for both which will be improved in future work.

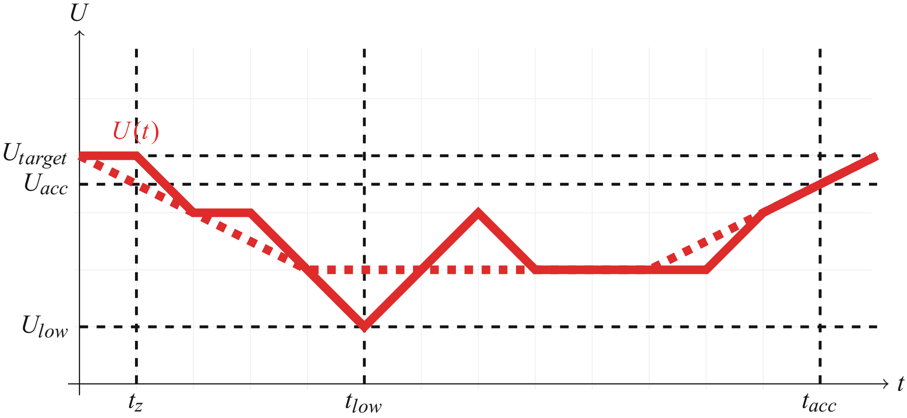


Fig. 15. Utility over time for one experiment with different sampling rates. Due to aliasing the robustness is different.

To solve this, we will extend the robustness into a transient and a permanent part. With that metric the permanent part would be the same for both graphs but the temporary component would be better for the dotted graph. As a drawback, this also would result in a more complicated metric.

Furthermore, Fig. 15 shows the same utility with different sampling rates. The robustness changes with the sampling rate if we just interpolate due to aliasing. However, this works if the sample points are already integrated as seen in the traffic example (see Sect. 6).

In future work, we will focus on quantitatively comparing similar mechanisms in different application scenarios. Also, we would like to extend the selection of integration limits for better comparison.

8 Conclusion

Complexity in information and communication technology is increasingly handled by systems themselves: Traditional design-time decisions are moved to run-time and into the responsibility of systems themselves. To do so, self-adaptation and self-organisation (SASO) mechanisms are developed that steer the required productive behaviour of the system and react on observable external effects. Besides reducing the administrative effort, this has the desired advantage that disturbances and attacks can be handled in an autonomous manner - we refer to this property of SASO systems as “robustness”. In this article, we argued that robustness is a key property to compare the efficiency and benefit of different SASO solutions, since it serves as one of the most important aspects to judge whether one specific solution is more beneficial than another.

In order to provide a universal technique to measure robustness in ICT systems in general (and SASO systems in particular), this article proposed to consider the utility degradation over time. By capturing the depth of the utility drop as well as the duration of the recovery phase, we determine a robustness value that is comparable among varying SASO mechanisms facing the same conditions. We discussed the state-of-the-art and explained that this approach overcomes the drawbacks of existing approaches, e.g. that they require too much internal information, measure only certain aspects (such as the time), are application-specific, or abstract the robustness too far (i.e., discretise the property into too few classes). In turn, the major advantage of the proposed method is the focus on externally measurable attributes – which allows for a generalised concept for comparing robustness. Conceptually, we further distinguish between a permanent part of robustness that is system inherent and a part that is generated by internal adaptation mechanisms. We demonstrated the expressiveness of the developed approach in terms of three case studies, i.e. from the desktop grid, the wireless sensor network, and the urban traffic management domains.

After analysing the behaviour of the proposed metric in terms of application scenarios, we identified limitations that need further research effort. In current and future work, we address these issues and develop a more sophisticated measurement framework. This extended framework will be able to take cases into account where a permanent drop in utility is recognised – the pre-defined least acceptable utility may not be achievable anymore. However, we still need to derive meaningful measurements. Furthermore, we will apply the metric to artificial and real-world use cases to highlight the generalised applicability of the developed technique.

References

2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems, Cambridge, MA, USA, 21–25 September 2015. IEEE Computer Society (2015)

2015 IEEE International Conference on Autonomic Computing, Grenoble, France, 7–10 July 2015. IEEE Computer Society (2015)

Barceló, J., Codina, E., Casas, J., Ferrer, J., García, D.: Microscopic traffic simulation: a tool for the design, analysis and evaluation of intelligent transport systems. J. Intell. Robot. Syst. **41**(2–3), 173–203 (2005)

Bazzan, A.L.C., Klügl, F. (eds.): Multi-Agent Systems for Traffic and Transportation Engineering. Idea Group Publishing, Hershey (2009)

Bellman, K.L., Tomforde, S., Würtz, R.P.: Interwoven systems: self-improving systems integration. In: Eighth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2014, London, United Kingdom, 8–12 September 2014, pp. 123–127 (2014)

Callaway, D.S., Newman, M.E., Strogatz, S.H., Watts, D.J.: Network robustness and fragility: percolation on random graphs. Phys. Rev. Lett. **85**(25), 5468 (2000)

Cámara, J., Correia, P., de Lemos, R., Vieira, M.: Empirical resilience evaluation of an architecture-based self-adaptive software system. In: Proceedings of 10th International ACM Sigsoft Conference on Quality of Software Architectures, pp. 63–72 (2014)

Chan, W.: Interaction metric of emergent behaviours in agent-based simulations. In: Proceedings of the Winter Simulation Conference, pp. 357–368 (2011)

Di Marzo Serugendo, G.: Robustness and dependability of self-organizing systems - a safety engineering perspective. In: Guerraoui, R., Petit, F. (eds.) SSS 2009. LNCS, vol. 5873, pp. 254–268. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-05118-0_18

Dinopoulou, V., Diakaki, C., Papageorgiou, M.: Applications of the urban traffic control strategy TUC. Eur. J. Oper. Res. **175**(3), 1652–1665 (2006)

Eberhardinger, B., Anders, G., Seebach, H., Siefert, F., Reif, W.: A research overview and evaluation of performance metrics for self-organization algorithms. In: 2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASO Workshops 2015, 21–25 September 2015Cambridge, MA, USA, pp. 122–127 (2015)

Edenhofer, S., Stifter, C., Jänen, U., Kantert, J., Tomforde, S., Hähner, J., Müller-Schloer, C.: An accusation-based strategy to handle undesirable behaviour in multi-agent systems. In: 2015 IEEE International Conference on Autonomic Computing, Grenoble, France, 7–10 July 2015, pp. 243–248 (2015)

Edenhofer, S., Tomforde, S., Kantert, J., Klejnowski, L., Bernard, Y., Hähner, J., Müller-Schloer, C.: Trust communities: an open, self-organised social infrastructure of autonomous agents. In: Reif, W., et al. (eds.) Trustworthy Open Self-Organising Systems. AS, pp. 127–152. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29201-4_5

Gronau, N.: Determinants of an appropriate degree of autonomy in a cyber-physical production system. In: Proceedings of 6th International Conference on Changeable, Agile, Reconfigurable, and Virtual Production, vol. 52, pp. 1–5 (2016)

Holzer, R., de Meer, H.: Quantitative modeling of self-organizing properties. In: Spyropoulos, T., Hummel, K.A. (eds.) IWSOS 2009. LNCS, vol. 5918, pp. 149–161. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10865-5_13

Holzer, R., de Meer, H.: Methods for approximations of quantitative measures in self-organizing systems. In: Bettstetter, C., Gershenson, C. (eds.) IWSOS 2011. LNCS, vol. 6557, pp. 1–15. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19167-1_1

Jalote, P.: Fault Tolerance in Distributed Systems. Prentice-Hall, Inc. (1994)

Kaddoum, E., Raibulet, C., Georgé, J.P., Picard, G., Gleizes, M.P.: Criteria for the evaluation of self-* systems. In: Proceedings of ICSE Workshop on Software Engineering for Adaptive and Self-Managing System, pp. 29–38 (2010)

Kantert, J., Edenhofer, S., Tomforde, S., Hähner, J., Müller-Schloer, C.: Normative control: controlling open distributed systems with autonomous entities. Trustworthy Open Self-Organising Systems. AS, pp. 89–126. Springer, Cham (2016a). https://doi.org/10.1007/978-3-319-29201-4_4

Kantert, J., Reinhard, F., von Zengen, G., Tomforde, S., Wolf, L., Müller-Schloer, C.: Combining trust and ETX to provide robust wireless sensor networks. In: Varbanescu, A.L. (ed.) Proceedings of the 29th International Conference on Architecture of Computing Systems Workshops (ARCSW 2016), pp. 1–7. VDE Verlag GmbH, Berlin, Offenbach, DE, Nuremberg, Germany (2016b). chap. 16

Kantert, J., Scharf, H., Edenhofer, S., Tomforde, S., Hähner, J., Müller-Schloer, C.: A graph analysis approach to detect attacks in multi-agent-systems at runtime. In: Proceedings of the Eighth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2014), pp. 80–89. IEEE, London, September 2014

Kantert, J., Tomforde, S., Müller-Schloer, C., Edenhofer, S., Sick, B.: Quantitative robustness - a generalised approach to compare the impact of disturbances in self-organising systems. In: Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017), vol. 1, pp. 39–50. SciTePress, Porto (2017). (Best Student Paper Award)

Kantert, J., et al.: Improving reliability and endurance using end-to-end trust in distributed low-power sensor networks. In: Pinho, L.M.P., Karl, W., Cohen, A., Brinkschulte, U. (eds.) ARCS 2015. LNCS, vol. 9017, pp. 135–145. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16086-3_11

Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE Comput. **36**(1), 41–50 (2003)

Klejnowski, L.: Trusted community: a novel multiagent organisation for open distributed systems. Ph.D. thesis, Leibniz Universität Hannover (2014). http://edok01.tib.uni-hannover.de/edoks/e01dh11/668667427.pdf

Menascé, D.A., Bennani, M.N., Ruan, H.: On the use of online analytic performance models, in self-managing and self-organizing computer systems. In: Babaoglu, O., Jelasity, M., Montresor, A., Fetzer, C., Leonardi, S., van Moorsel, A., van Steen, M. (eds.) SELF-STAR 2004. LNCS, vol. 3460, pp. 128–142. Springer, Heidelberg (2005). https://doi.org/10.1007/11428589_9

Nafz, F., Seebach, H., Steghöfer, J.P., Anders, G., Reif, W.: Constraining Self-organisation Through Corridors of Correct Behaviour: The Restore Invariant Approach. In: Müller-Schloer, C., Schmeck, H., Ungerer, T. (eds.) Organic Computing - A Paradigm Shift for Complex Systems. Autonomic Systems, pp. 79–93. Birkhäuse verlag, Basel (2011)

Nimis, J., Lockemann, P.C.: Robust multi-agent systems: the transactional conversation approach. In: First International Workshop on Safety and Security in Multiagent Systems (SASEMAS 2004), pp. 73–84 (2004)

Prothmann, H., Tomforde, S., Lyda, J., Branke, J., Hähner, J., Müller-Schloer, C., Schmeck, H.: Self-organised routing for road networks. In: Kuipers, F.A., Heegaard, P.E. (eds.) IWSOS 2012. LNCS, vol. 7166, pp. 48–59. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28583-7_5

Prothmann, H., Tomforde, S., Branke, J., Hähner, J., Müller-Schloer, C., Schmeck, H.: Organic traffic control. In: Organic Computing - A Paradigm Shift for Complex Systems, pp. 431–446. Birkhäuser Verlag (2011)

Schmeck, H., Müller-Schloer, C., Çakar, E., Mnif, M., Richter, U.: Adaptivity and self-organisation in organic computing systems. ACM Trans. Auton. Adapt. Syst. (TAAS) **5**(3), 1–32 (2010)

Scholl, A., et al.: Robuste planung und optimierung: Grundlagen, konzepte und methoden; experimentelle untersuchungen. Tech. rep., Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL) (2000)

Schwind, N., Okimoto, T., Inoue, K., Chan, H., Ribeiro, T., Minami, K., Maruyama, H.: Systems resilience: a challenge problem for dynamic constraint-based agent systems. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems, pp. 785–788. International Foundation for Autonomous Agents and Multiagent Systems (2013)

Slotine, J.J.E., Li, W., et al.: Applied Nonlinear Control, vol. 199. Prentice-Hall, Englewood Cliffs (1991)

Taguchi, G.: Robust technology development. Mech. Eng. CIME **115**(3), 60–63 (1993)

Tanenbaum, A.S.: Computer Networks, 4th edn. Pearson Education (2002)

Tomforde, S., Hähner, J., Seebach, H., Reif, W., Sick, B., Wacker, A., Scholtes, I.: Engineering and mastering interwoven systems. In: ARCS 2014–27th International Conference on Architecture of Computing Systems, Workshop Proceedings, 25–28 February 2014, Luebeck, Germany, pp. 1–8. University of Luebeck, Institute of Computer Engineering (2014)

Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Mnif, M., Müller-Schloer, C., Richter, U., Schmeck, H.: Observation and control of organic systems. In: Organic Computing - A Paradigm Shift for Complex Systems, pp. 325–338. Birkhäuser Verlag (2011)

Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Müller-Schloer, C., Schmeck, H.: Possibilities and limitations of decentralised traffic control systems. In: 2010 IEEE World Congress on Computational Intelligence (IEEE WCCI 2010), pp. 3298–3306. IEEE (2010)

Tomforde, S., Sick, B., Müller-Schloer, C.: Organic Computing in the Spotlight, January 2017. http://arxiv.org/abs/1701.08125

Transportation Research Board: Highway capacity manual. Technical report, National Research Council, Washington D.C., US (2000)

Parunak, H.V.D., Brueckner, S.: Entropy and self-organization in multi-agent systems. In: Proceedings of the Fifth International Conference on Autonomous Agents (AGENTS 2001), pp. 124–130. ACM, New York (2001)

Wedde, H.F., Lehnhoff, S., et al.: Highly dynamic and adaptive traffic congestion avoidance in real-time inspired by honey bee behavior. In: Holleczek, P., Vogel-Heuser, B. (eds.) Mobilität und Echtzeit - Fachtagung der GI-Fachgruppe Echtzeitsysteme, pp. 21–31. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74837-3_3

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., Alexander, R.: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard), March 2012. http://www.ietf.org/rfc/rfc6550.txt

Wooldridge, M.: An Introduction to Multiagent systems. Wiley (2009)

Analysis of Perceived Helpfulness in Adaptive Autonomous Agent Populations

Mirgita Frasheri(✉), Baran Çürüklü, and Mikael Ekström

Mälardalen University, Västerås, Sweden
{mirgita.frasheri,baran.curuklu,mikael.ekstrom}@mdh.se

Abstract. Adaptive autonomy allows agents to change their autonomy levels based on circumstances, e.g. when they decide to rely upon one another for completing tasks. In this paper, two configurations of agent models for adaptive autonomy are discussed. In the former configuration, the adaptive autonomous behavior is modeled through the willingness of an agent to assist others in the population. An agent that completes a high number of tasks, with respect to a predefined threshold, increases its willingness, and vice-versa. Results show that, agents complete more tasks when they are willing to give help, however the need for such help needs to be low. Agents configured to be helpful will perform well among alike agents. The second configuration extends the first by adding the willingness to ask for help. Furthermore, the perceived helpfulness of the population and of the agent asking for help are used as input in the calculation of the willingness to give help. Simulations were run for three different scenarios. (i) A helpful agent which operates among an unhelpful population, (ii) an unhelpful agent which operates in a helpful populations, and (iii) a population split in half between helpful and unhelpful agents. Results for all scenarios show that, by using such trait of the population in the calculation of willingness and given enough interactions, helpful agents can control the degree of exploitation by unhelpful agents.

Keywords: Adaptive autonomy · Collaborative agents
Multi-agent systems

1 Introduction

Adaptive autonomous (AA) agents are software agents which are able to decide on whether to be more or less autonomous, with respect to some task, given specific circumstances. Autonomy can both refer to the behavior of an agent, in any given context, as well as the agent's relationship with other entities. The latter can include other AA agents, non AA agents, or human operators that operate in a nearby environment (physical or virtual), such that communication between them is possible. The decision to change the autonomy level could also lie on the human operator, or be a result of the cooperation between human and agent. Thus, adaptive autonomy is one of many concepts that involves the

© Springer International Publishing AG, part of Springer Nature 2018
N. T. Nguyen et al. (Eds.): TCCI XXVIII, LNCS 10780, pp. 221–252, 2018.
https://doi.org/10.1007/978-3-319-78301-7_10

change of the autonomy level of an agent. A non-exhaustive list of theories and definitions, in addition to AA is as follows: adjustable autonomy, mixed-initiative interaction, sliding autonomy, collaborative control and so on.

Adjustable autonomy refers to a system in which the human is the one who makes the decision with respect to the autonomy of an agent [1]. Nonetheless, it has also been used as a generic term for the different means in which decision-making regarding autonomy could be shared between human and agent [2]. In mixed-initiative interaction, agent and human are both able to make a decision, depending as well on the circumstances [1]. Collaborative control [3] is an early approach, which departed from a classical view of human/master - agent/slave, into one in which both were peers. Any inconsistencies between them were resolved through dialogue. Nonetheless, the human was the one who would set the global goals for the agent. Sliding autonomy represents another approach in which two modes (full autonomy and tele-operation) could be switched on the task level [4]. Consequently, an operator is able to conduct some tasks, whilst the system performs autonomously for others, depending on the circumstances. Agent autonomy has been studied extensively in the literature. The 10 levels of autonomy have been proposed by Parasuraman *et al.* [5] (Table 1). Another scheme is through the dimensions of self-sufficiency, i.e. being able to do a task without outside assistance, and self-directedness, i.e. being able to choose one's own goals. Johnson *et al.* [2] refer to them as the descriptive and prescriptive dimensions of autonomy, respectively. Furthermore, they add a third dimension, that of inter-dependencies between team-mates (either agent or human). The former could be either hard, i.e. necessary for the successful outcome of a task, or soft, i.e. not necessary, however, could improve on the performance of a task. Castelfranchi defines autonomy using dependence theory [6], i.e. if an agent a_i lacks any means (e.g. ability, knowledge, or external resources) to perform a task t and relies/depends on another agent a_j for its provision, then a_i is not autonomous from a_j with respect to t. Moreover, this kind of autonomy/non autonomy has a social nature, and is to be distinguished from autonomy from the environment, or in other words autonomy with respect to how to react to incoming stimuli. This paper assumes [6], thus allows agents to adapt their autonomy by deciding on whether to depend on each other. Section 2 provides more information on the research conducted in the field of AA and alike concepts. It also puts the work presented in this paper into perspective with respect to the literature.

A model for an AA agent has been proposed previously [7], in which adaptation is modeled through the willingness of agents to give assistance to each other. The agent's own performance – calculated as the number of tasks completed over tasks attempted – influences the willingness to give assistance. If the performance is high, the agent will be more willing to help, the opposite also being true. This is **Configuration 1** ($C1$) of the agent model. $C1$, alongside dedicated simulations are shortly described in this paper, specifically in Sects. 3.1–3.3 and 5.1. Subsequently, the AA agent architecture was extended through the incorporation of another behavior: willingness to ask for help (Sect. 3.4). Thus, the current

Table 1. The 10 levels of autonomy proposed by Parasuraman *et al.*

HIGH	10. The computer decides everything, acts autonomously, ignoring the human
	9. informs the human only if it, the computer, decides to
	8. informs the human only if asked, or
	7. executes automatically, then necessarily informs the human, and
	6. allows the human a restricted time to veto before automatic execution, or
	5. executes that suggestion if the human approves, or
	4. suggest an alternative
	3. narrows the selection down to a few, or
	2. The computer offers a complete set of decision/action alternatives, or
LOW	1. The computer offers no assistance: human must take all the decisions/actions

model incorporates both directions of communication with respect to allowing agents to help each other. This is **Configuration 2** ($C2$).

The next step is to consider the interactions from a population perspective. The characteristic of the population under study is the perceived helpfulness of its individuals taken as a group. It is defined by the number of times in which agents have been willing to assist an agent a_i, over the total number of requests for help made by a_i. Each agent in the group can estimate the perceived helpfulness (i) of the population, and (ii) of individual agents. Both measures are combined and used in the calculation of the willingness to give help, and extend $C2$ (Sect. 4). The perceived helpfulness of the population is referred throughout the text as part of *agent culture*. The main objective is to analyse how the (most) helpful AA agents can avoid extensive exploitation of their resources by other agents. The corresponding simulations and results are summarized in Sect. 5.2. Three hypotheses are evaluated.

Hypothesis 1 (H1): *Exploitation of an agent by its peers can be lowered by considering the helpfulness of the population as a whole as well as that of an individual agent in the calculation of willingness to give help.*

Hypothesis 2 (H2): *An agent population can adapt to an agent configured so as to exploit by considering the helpfulness of the population as a whole as well as that of an individual agent in the calculation of the willingness to give help. Moreover, the efficiency of such isolation is disproportional to the population size.*

Hypothesis 3 (H3): *Agents in a mixed population, where half of the population is helpful and the other half is otherwise, can reduce exploitation while still helping each other by considering the helpfulness of the population as a whole as well as that of an individual agent in the calculation of the willingness to give help.*

Finally, the paper concludes with a discussion and reflections for future work (Sect. 6), and conclusions (Sect. 7).

2 Related Work

Agent autonomy is an extensively discussed topic in the literature. A close examination of the related work points at six (6) main directions of research: (i) design of user interfaces which aid human/robot(agent) collaboration, (ii) specific algorithms that allow for autonomy levels of agents/robots to be changed such as Markov Decision Processes, (iii) design of policy systems for the regulation of agent behavior, (iv) works that aim at comparing different schemes for changing autonomy levels and works that motivate the need for such change, (v) design methodologies for the creation of systems that support inter-dependencies between systems, (vi) general architectures and frameworks. The next paragraphs provide a compact description of relevant literature in each of the mentioned directions. Furthermore, the research conducted in this paper is put into perspective with the existing work found in the literature.

(i) User interfaces are used as means to allow both agent and human to monitor each other, and consequently change autonomy levels if perceived necessary. The initiation of change can come from both sides. A system able to capture the user's skill, can change its autonomy accordingly [8]. In this case skills are of a navigational, manipulation (gripping), and multi-robot coordination nature. On the other hand, a human operator can have the flexibility to command a robot at several levels such as: low-level control, way-point control, high-level control (sending goals like "bring the can of coke") [9]. Other types of interfaces are aimed at aiding a human to monitor and control a group of robots, which may need only occasional support [10,11]. Other work is specific to navigational issues, in which a human assists path planning of UAVs (unmanned aerial vehicles) by providing spatial and temporal constraints [12]. The 3 T agent architecture [13] has been extended to allow for the human in the loop of the decision-making of the agent [14]. The addition enables the system to keep track of what the human does, so as not to lose the common picture between human/system. Overall, the challenge for these interfaces is to allow for the common ground not to be lost and be accessible by all parties [15,16]. Moreover, an important consideration to be made has to do with how much autonomy the agent/robot is intended to have [17].

(ii) Attributes such as task urgency and dedication level to the organization haven been used to guide the agent's reasoning with respect to when to take more initiative (increase its autonomy) [18]. Autonomy levels can also be changed at the task level, i.e. one task needs tele-operation from the human, whereas another can be conducted autonomously [4]. Furthermore, several task allocation algorithms have been proposed. In one, tasks are mapped to agents, and the human is able to accept/reject such mapping and trigger task allocation from the start [19]. Others categorize tasks in two groups: tasks which the agent can perform autonomously, and tasks that need human assistance [20]. The classification influences algorithm design. Colored Petri Nets are used in the formalization of team plans and addition of interrupt mechanisms, which allow the human to intervene in case of need [21]. Markov Decision Processes (MDPs) are employed to map help requests from agents with available humans – assuming that agents detect when they are in trouble [22]. In other threat recognition and target identification applications, the system is implemented to query a human when it fails [23].

(iii) Regulatory systems (e.g. policies) are discussed in the context of regulating agent behavior, due to bringing predictability and thus coordination [24]. An example is the Kaa system [25] which extends the KAoS policy system, by introducing a central agent (the Kaa) to override and adjust policies during run-time. If Kaa cannot reach a decision, then the human is introduced in the loop. Another approach involves the implementation of transfer-of-control strategies (through MDPs), which specify how control should be transferred between human and agents [26]. This has been applied in the E-elves platform (personal assistant agents) which ran at the University of Southern California.

(iv) The ability to change autonomy levels is considered a desirable features of systems, which can allow them to operate in human-team like fashion [27]. Scenarios with and without the ability to change autonomy have been compared [28,29]. Decision-making frameworks such as master/slave, peer-2-peer and locally autonomous are dynamically shifted to show the superiority compared to static autonomy. However, the authors use data from previous experiments to apply the right decision framework for each environmental condition; there is no reasoning embedded in the agents. Different implementations of dynamic autonomy have been compared, which are adaptive autonomy (agents change their own autonomy), mixed-initiative interaction (both human and agent are able to change autonomy), and adjustable autonomy (the human is able to change the autonomy) [1]. In their simulations, mixed-initiative interaction performs better in terms of victims identified in search and rescue simulation scenario.

(v) Jonson *et al.* have argued the need for the analysis of inter-dependencies between systems, and its use in the design phase [2]. Moreover, they have proposed one such methodology [30], namely Co-active Design. This method includes the following steps. (i) Inter-dependencies in the system are identified. (ii) Mechanisms are designed to address each inter-dependency. (iii) The effects of these mechanisms on present inter-dependent relationships are analyzed. The aim is to make automation a team-player. In this respect several challenges have been identified [31] such as: basic compact, adequate models, predictability, directability, revealing status and intention, goal negotiation, collaboration, attention management, and cost control.

(vi) The agent architecture STEAM [32] has extended the Soar agent [33] to include support for teamwork. Team operators – reactive team plans – are introduced. These are an addition to the agent's plans that do not require teamwork. The solution includes a synchronization protocol so that agents can coordinate with respect to team plans. The DEFACTO framework [34] aims at providing support for transfers of control in continuous time,

> resolving human-agent inconsistencies, and making actions interruptible for real-time systems. Team THOR's Entry in the DARPA Robotics Challenge [35] brings forward a motion framework for a humanoid robot which allows for low-level control, scripted autonomy (i.e. invocation of robot movements by calling predefined scripts), and enables issuing high-level commands.

The research discussed in this paper fits mostly within (ii) and (vi). On one hand, algorithms are being developed to allow agents to assist and ask each other for assistance during their run-time. There are no classifications of tasks that either need assistance or not. In principle, an agent might require help for any task, due to changing circumstances. Assume an agent a_i which at time T_1 is able to perform task t. The same agent, at time T_2 might not be able anymore to continue on its own. One reason could be that, its battery levels have gone down. On the other hand, these algorithms fit within a general agent architecture, which models how an agent executes during its run-time.

3 Agent Model

This section describes configurations 1 and 2 of the agent model.

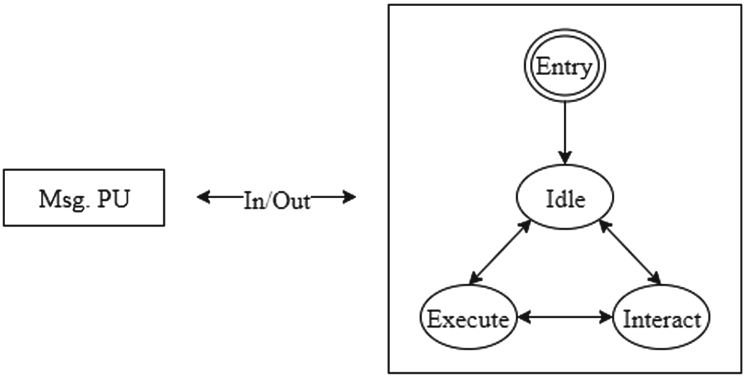


Fig. 1. C1 agent model with three states [7]. The Msg. PU represents the module in which messages coming from other agents are handled

3.1 Early Work (C1)

The agent model proposed previously [7] consists of the three (3) states, *interact*, *execute*, and *idle* (Fig. 1). All agents in the population have a willingness to give assistance to each other. This concept is represented by a probability value, which defines the likelihood for such an event to occur. Assume an agent a_i which is in either the states of *idle* or *execute*. If the agent is in *execute*, it means that it is already dedicated to a task. Agent a_i can be in either *idle* or *execute* when a request for assistance is received, and will switch immediately to the

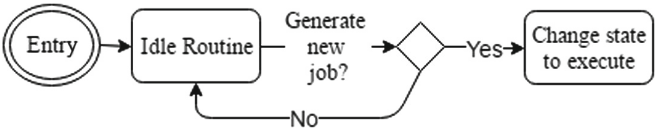


Fig. 2. Flowchart for the *idle* state [7].

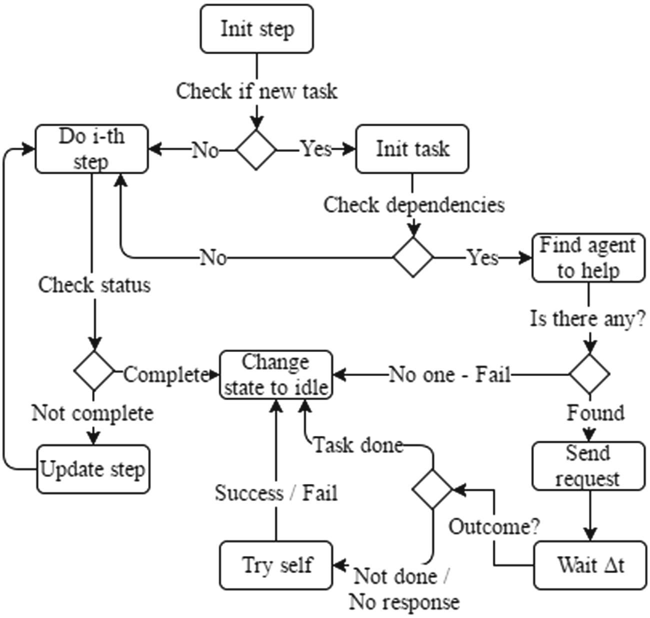


Fig. 3. Flowchart for the *execute* state [7].

interact state, where the decision of whether to accept the request will be made. If a_i accepts the request, it will switch to *execute* with the new corresponding task, whilst the old one will be dropped for good. Otherwise, it will switch to its previous state, i.e. either *idle* or *execute* and continue with the old task.

An agent always starts its operation in the *idle* state. In this state, the agent is not dedicated to any task or goal. Nonetheless, a task could be generated with a probability P (Fig. 2). This task is picked from a list of tasks which the agent is able to do (specified before runtime). When a task is generated, or a request for help is accepted, the agent switches to *execute*. The assumption is made that if the agent is not interrupted, it will always complete its task successfully (Fig. 3). At the beginning of each task, an agent will check for dependencies on other tasks. In the case of C1, dependencies are assumed to be fixed and known before-hand by all agents. If there is any, it will issue a help request to some known agent a_j which is able to perform the task. Then it will wait for a finite amount of time (constant determined before runtime and equal to all agents) for a response from a_j. If there is no response, the agent will give up on waiting, update the history of interactions with a_j, and attempt to do the task by itself with a lower probability of success. There are other options which could be implemented as well. (i) The agent can first try by itself, and if it fails, asks another for help. (ii) After a_j fails, the agent attempts with another agent a_k. When a task is finished, the agent returns to *idle*.

Agents keep track of the outcomes of the interactions with each other. As a result, they are able to compute the perceived willingness to help and expertise with respect to some task of every other agent. An agent a_i will select another agent a_j to ask for help based on a_j's helpfulness in the past, i.e. based on a_j's perceived willingness to help. During its operation, the agent keeps track of the outcomes for each interaction, which is used to determine helpfulness. It is important to note the difference between the perceived helpfulness and expertise. The former is an indicator of how much another agent has been willing to help, whereas the latter captures the actual success rate of the agent that has been trying to help. As a result, the perceived willingness is a more optimistic measure in which to judge other agents.

When the agent gets a request from another agent for help, then it will switch to the *interact* state which cannot be interrupted, i.e. it can be considered as an atomic step (Fig. 4). This has the implication that the requests will be processed one at a time in a FIFO manner. There are two possible outcomes from the *interact* state. The agent can drop the past activity and go into *execute* with a new task, or it discards the requests and continues with what it was doing before receiving the request. The willingness to give help is the determining factor that shapes agent behavior. The agent performance will in turn influence the willingness to give help in the following way. If the agent calculates that it has dropped too many tasks, then its willingness will decrease. On the other hand, if it has completed most of the tasks it has attempted, then its willingness will increase.

3.2 Interactions Between Agents

In order to resolve the dependencies between them, agents need to interact with each other. This means that, if an agent a_i doing a task t identifies that it needs to depend on a_j to complete t, then a_i will need to interact with a_j. Dependencies themselves could be known in advance (as assumed for the agent described in Sect. 3.1) or could arise during runtime. Moreover, they can arise at the beginning or during the execution of a task t.

There are several types of possible interaction between agents:

1. Non-committal interaction. Agents could broadcast certain messages to others in the vicinity. These messages can contain different kinds of information related to e.g. identity, offered services, warnings ("There is fire in corridor x", "path from x_1 to x_2 is blocked"). Other agents are able to accept or disregard them. Nonetheless, no dialogue is being established by the involved parties. This means that the agent sending a broadcast does not expect any

reply or commitment from others. In this work, this interaction is used by agents to make themselves known to each other. In principle, other agents could be able to evaluate the trustworthiness of the broadcasting agent by examining the following: (i) is the information useful, and (ii) is it true?

2. One-to-one dialogue. Agent a_i misses specific information, and queries a_j. In this case, a one-to-one dialogue is being established, in which one party expects a reply from the other, so that it is able to fill its knowledge gaps. The validity of the information a_j provides could be evaluated, as well as its perceived helpfulness to a_i.
3. One-to-one delegation. Similarly to one-to-one dialogue, a kind of dialogue is established in this case as well, in the form of a request to complete a task. This means that a_i will ask a_j to perform an activity on which $a_i's$ success with respect to a task depends. In general, a_i could be still able to succeed by itself, but with a lower probability. a_i is able to evaluate the behavior of a_j based on (i) its perceived helpfulness and (ii) shown expertise. a_j as well will perform an evaluation in order to determine whether to assist a_i. This type interaction is as well implemented in this paper.
4. One-to-many dialogue/delegation. There are two ways to interpret this scenario. (i) There could be a chain of one-to-one interaction which emerges, e.g. a_i asks a_j, which asks a_k and so on. In this paper, such kind of chains can emerge. (ii) An agent can start parallel interactions with a number of other agents (B, C, etc.) by asking each of them to perform some specific subtask.

Johnson *et al.* [2] consider in their work soft and hard interdependencies between agents. Each level of interaction described in the previous paragraph could refer to either depending on the concrete scenario. For instance, a noncommittal broadcast message could contain an alarm (e.g. "There is fire in x corridor") and be decisive for the outcome of a task (and even well-being of the agents). Thus it represents a hard interdependence. However, an ordinary informational message (e.g. "path from x_1 to x_2 is blocked") if disregarded could only delay the execution of some task without hindering its success. As a result, it can be considered as a soft interdependence. Furthermore, the difference with Barber *et al.* [29] is that the decision to assist another agent lies on the agent itself. This means that, a_i can ask to delegate a task to a_j, and a_j reasons and decides whether to accept such delegation.

3.3 Agent Organization and Autonomy

Agent organization will have an impact on how autonomy is shaped for each individual agent that makes up the population. There are two possibilities. (i) There is a hierarchy between agents which could be predefined or could emerge (e.g Barber *et al.* [29] consider how environmental conditions could be used to evaluate which hierarchy fits best a specific scenario). (ii) Agents are peers with each other.

In the first case, an agent a_i which is a superior of a_j is able to delegate to a_j any task it sees fit with some assurance that a_j will comply. Delegating to a_j

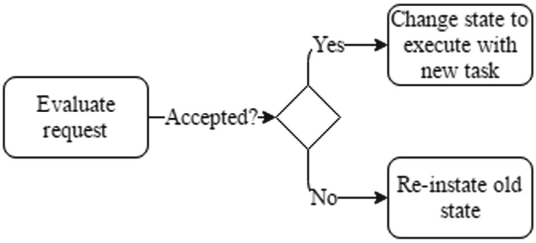


Fig. 4. Flowchart for the *interact* state [7].

does not necessarily mean that a_i cannot perform the task by itself. It can very reasonably be assumed that a_i is able to perform the task but simply prefers to conserve its resources, and ask a_j instead. a_j on the other hand either has some freedom in which it can refute to obey a_i (e.g. what a_i asks endangers a_j in ways that may or may not have been foreseen by a_i) or it does not and it will always have to comply. As a result, in general a_j will be dependent on the will of a_i, given that the power relations between them hold.

In the second case, a_i and a_j are peers, thus no power relations between them can be assumed. If a_i needs help, it will make a request to a_j, which in turn will decide based on its willingness to give help whether to assist a_i. As such, it is a_i which depends on the will of a_j. Nevertheless other factors might come into play. If a_i has been helping a_j in the past, then the latter could be more inclined to return the favor, thus becoming easier to interfere with. Furthermore, the motives of a_j might not be genuine (help a_i because it has helped me), but they could in fact be more along the lines of: help a_i so it can continue helping me in the future.

Note that, some form of dependence between agents is present in both scenarios. Moreover, dependence always constitutes a risk [36]. Even when there are power relations, an agent choosing to delegate to another, is choosing to depend, and thus is giving away some of its autonomy. The agent that delegates might be able to do the task by itself, if another agent fails. However, if the output of a task is expected within a certain time, then a delay could mean failure. On the other hand, if the agent cannot perform the task by itself, then it is even more dependent on the agent it asks for help. Therefore, the level of autonomy cannot always be well defined and can be blurred. In this paper, agents are assumed to be peers.

3.4 The AA Agent (C2)

The agent model described in Sect. 3.1 has been extended with respect to two dimensions. (i) Two more supporting states have been added (Fig. 5), the *regenerate* and *out_of_order* states respectively. If the agent reaches critical levels of

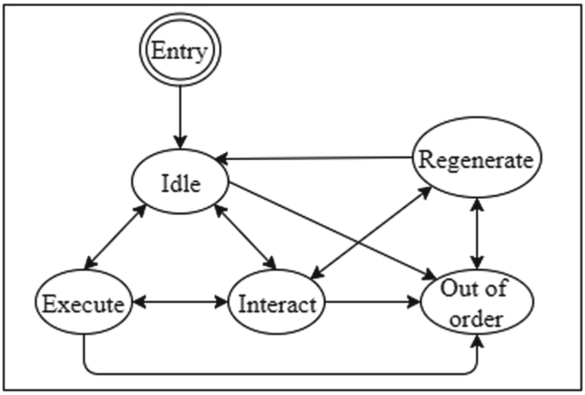


Fig. 5. Agent model extended to include two more states: *regenerate* and *out_of_order*

battery (the start-up energy level, and the critical level are arbitrarily specified before runtime) it will switch to the *out_of_order* state, and immediately from that state it will switch to *regenerate* where the recharge process is simulated. Next, the agent will switch to either *idle* or *interact* (if there are any requests pending). The agent can go to *out_of_order* from any other state (in principle from *regenerate* as well). From the implementation perspective, ROS services (for one-to-one interaction) have been switched with the ROS action server mechanism. The limitation of the current implementation is that the execution of a task in the *execute* state is simulated as a singular step, by pausing the system for a finite amount of time (which represents the completion time for a task). As a result, an agent reasons on whether it should ask for help at the beginning of each task.

(ii) The agent's adaptive autonomous behavior is shaped by the willingness to interact which is composed of the willingness to give help (δ), and the willingness to ask for help (γ). The willingness to ask for help represents the probability that an agent will ask another for assistance given its current circumstances (in this case, dependencies are assumed to rise during runtime – in contrast to C1). Moreover, the factors assumed to influence each facet of the willingness to interact have been analyzed, and a corresponding computational model has been proposed from which δ and γ are calculated. The calculation is done according to Algorithms 1 and 2.a. The notation used in both has the following meaning: b - battery, e - equipment, k - knowledge, a_i - abilities, n_t - tools, μ - agent performance, e_R - environmental risk, t_p - task progress/trade-off, a_R - perceived agent risk (complementary to perceived helpfulness). The notations with the subscript t refer to abilities, resources needed by the task. In the cases when it misses it refers to what the agent has at its disposition.

Algorithm 1. Agent's reasoning process on when to ask for help

```
procedure REASONING ON ASKING FOR HELP($b, e, k, a, t, \mu, e_R, t\_p, a\_R$)
    $\gamma \leftarrow \gamma_0$
    if $b - b_t < b_{min}$ or $e_t \not\subset e$ or $k_t \not\subset k$ or $a_t \not\subset a$ or $t_t \not\subset t_t$ then ▷ Consider internal resources
        $\gamma \leftarrow 1$
        return $\gamma$
    else
        $\gamma \leftarrow \gamma - 5\Delta\gamma$
        if $e_R$ increase then                                  ▷ Consider environment risk
            $\gamma \leftarrow \gamma + \Delta\gamma$
        else
            $\gamma \leftarrow \gamma - \Delta\gamma$
        if $a\_R$ increase then                                 ▷ Consider agent risk
            $\gamma \leftarrow \gamma - \Delta\gamma$
        else
            $\gamma \leftarrow \gamma + \Delta\gamma$
        if $\mu$ increase then                                  ▷ Consider own performance
            $\gamma \leftarrow \gamma - \Delta\gamma$
        else
            $\gamma \leftarrow \gamma + \Delta\gamma$
        if $t\_p$ good then                                     ▷ Consider task progress
            $\gamma \leftarrow \gamma - \Delta\gamma$
        else
            $\gamma \leftarrow \gamma + \Delta\gamma$
    return $\gamma$
```

3.5 ROS

The Robot operating system (ROS) [37] serves the role of a middleware by implementing different communication mechanisms such as: (i) publish/subscribe, (ii) services, and (iii) action servers. A ROS executable is called a node. Nodes publish and subscribe to named buses referred to as topics. Services are point to point communications between one client and one server. Action servers improve on services by allowing the server node to send progress feedback to the client after an initial server call. Nodes are able to find each other through the ROS master, which supplies naming and registration services.

In this work, in both C1 and C2 configurations, agents are composed of two ROS nodes. The main agent node contains the logic, and the message processing unit node handles published data from other agents. Nodes are written in the python language, which is supported in ROS (alongside other supported languages such as C++ and Java). In C1, the services are used to implement the one-to-one delegation interaction. In C2, service calls are replaced with action

Algorithm 2.a. Agent's reasoning process on when to give help

```
procedure REASONING ON GIVING HELP($b, e, k, a, t, \mu, e_R, t\_p, a_R$)
    $\delta \leftarrow \delta_0$
    if $b - b_t < b_{min}$ then                          ▷ Consider internal resources
        $\delta \leftarrow 0$
        return $\delta$
    else
        $\delta \leftarrow \delta + \Delta\delta$
        if $e_t \not\subset e$ then                      ▷ Consider internal resources
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow \delta + \Delta\delta$
        if $k_t \not\subset k$ then                      ▷ Consider internal resources
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow \delta + \Delta\delta$
        if $a_t \not\subset a$ then                      ▷ Consider internal resources
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow \delta + \Delta\delta$
        if $t_t \not\subset t$ then                      ▷ Consider external resources
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow \delta + \Delta\delta$
        if $e_R$ increase then                           ▷ Consider environment risk
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow \delta + \Delta\delta$
        if $a_R$ increase then                           ▷ Consider agent risk
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow \delta + \Delta\delta$
        if $\mu$ increase then                           ▷ Consider own performance
            $\delta \leftarrow \delta + \Delta\delta$
        else
            $\delta \leftarrow \delta - \Delta\delta$
        if $t\_p$ good then                              ▷ Consider task progress
            $\delta \leftarrow \delta + \Delta\delta$
        else
            $\delta \leftarrow \delta - \Delta\delta$
    return $\delta$
```

server calls. At present there is no substantial difference between the two implementations in the code. Nevertheless, the transition was made in order to allow more flexibility for future development.

Algorithm 2.b. Reasoning on δ with perceived helpfulness

```
procedure ADDITIONAL REASONING ON $\delta(a\_R, a\_S, C)$
    $\delta \leftarrow \delta_p$                                  ▷ $\delta_p$ calculated in Algorithm 2.a
    if $a_R >= 0.5$ and $C <= 0.5$ then
        $\delta \leftarrow LOW$
    else if $(a_R < 0.5$ and $C <= 0.5)$ or $(a_R < 0.5$ and $C > 0.5)$ then
        $\delta \leftarrow \delta + k * \Delta step$
    else if $a_R >= 0.5$ and $C > 0.5$ then
        if $a\_S < 3$ then
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow LOW$
```

4 Perceived Helpfulness

This paper extends the ideas presented and discussed in previous work [7] by including in the agent's reasoning, characteristics of the population in which the agent operates. These additions take place in Configuration 2. One such characteristic is the perceived helpfulness of the whole population of agents (or the part of the population with which an agent is able to communicate and interact with). This is the target in this work. The perceived helpfulness of the population is referred to as part of its *culture*. Other facets can be part of agents' culture, such as perceived willingness to ask for assistance. However, the latter is not treated here.

Assume an agent a_i found in an environment among other agents (no particular hierarchy is imposed). During its operation, a_i sends to and receives from others help requests. Continuously, upon each request, a_i has to decide whether to help or turn down the request. Moreover, a_i will need to depend on other agent on particular circumstances. Previously, it has been showed that an agent configuration with high willingness to give help, in situations where dependencies are low (i.e. it asks for help in few cases), results in the highest number of tasks completed (performance measure) for the whole population [7]. It is reasonable to think that such a population will succeed with respect to the number of tasks completed.

Note that, it is not always possible to assume that an agent will find itself among agents configured in the same manner. As such, an agent a_i with $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, operating around a population of agents configured oppositely, i.e. $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$, will be exploited and will not get the assistance it needs from its selfish peers. Therefore, a_i needs to take into account the behavior of individual agents as well as the whole population, so that it can conserve its resources when faced with exploiters.

In order to aid the agent such that it is not exploited, Algorithm 2.a is extended through Algorithm 2.b. The main behavior expected to be produced by this extension is the following. If an agent a_i finds itself in a selfish society then, if the perceived helpfulness of the particular agent a_j asking for help is below a given threshold, a_i will lower its δ to a value as low as 0.1, which means that if it will help with a probability 0.1. Conversely, if a_j's perceived helpfulness is above a given threshold, then independently of the population, a_i will increase its δ by $k * \Delta step$. The reason behind such choice is that, a_i should not penalize other helpful agents, even though the agent population it resides in is selfish. Finally, if the agent population is helpful, i.e. if the perceived helpfulness is above a given threshold, although a_j's perceived helpfulness is low, then a_i penalizes a_j only after a certain number of unhelpful interactions. Thus, a_j is given the chance to change its behavior so as to become more helpful.

5 Simulations

This section describes the simulation setups and results for configurations 1 and 2 of the agent model.

5.1 Setup and Results for C1

C1 Setup. The agents are implemented as ROS nodes [37] and interact through the publish/subscribe mechanisms and services. The simulation is limited to three types of interactions, the non-committal broadcast (implemented through ROS publish/subscribe), and the one-to-one delegation (implemented through ROS services), and the emergent chain of one-to-one delegations. The agents make themselves continuously known to each other by broadcasting their identity and the list of tasks the are able to execute. Note that, agents are not assumed to have the same global goals. Thus each one has its own objective, however can put its capabilities to the service of others if the need arises.

Simulations were setup in order to investigate the utility of the agent population. Utility is measured on two levels, (i) the degree of completion of dependent tasks (CD), and (ii) the degree of dropped tasks (DD). Dependent tasks are tasks which depend on other tasks in order to have a higher chance for a successful outcome. As such, CD is calculated by dividing the number of dependent tasks completed over the number of dependent tasks attempted (Eq. 1).

$$CD = \frac{Depend\ Tasks\ Completed}{Depend\ Tasks\ Attempted} \tag{1}$$

DD is calculated by dividing the number of dropped tasks over the number of attempted tasks (Eq. 2).

$$DD = \frac{Tasks\ not\ Completed}{Tasks\ Attempted} \tag{2}$$

Two parameters were manipulated in the tests, the dependency degree between tasks, and the willingness to give help (δ). The former refers to the percentage of tasks (in a given set) that depend on other tasks for a higher probability of success. The latter represents the probability that an agent will assist another agent when requested.

In the simulations there are 10 tasks that are defined as abstract entities. Each task is simulated as a *for loop* which runs for a specific amount of iterations. The list of tasks an agent is able to do is a subset of 10 tasks. More than one agent is able to execute the same task. As a result, there is diversity with respect to whom an agent can ask for help. On each run of the simulation, the same agent provides the same set of tasks. Moreover, the dependencies between tasks are given before-hand. In this work, the dependence is limited to one task, as opposed to many. An agent asking for help will wait $\Delta t = 60\,\text{s}$ before dropping the request. This value is given before runtime and is assumed equal to all agents.

Three sets of simulation runs were conducted. Each set includes three independent runs for the same fixed parameters: population size (*popsize*), dependency degree (*depdeg*), and Greek delta δ. Simulation set 1: δ is static, i.e. does not change throughout one simulation run, and *popsize* takes the values 10 and 30 for separate runs. The parameter δ takes its values in the segment $[0.0, 0.25, 0.5, 0.75, 1.0]$. *Depdeg* takes values in $[10\%, 25\%, 50\%, 75\%, 100\%]$. Simulations were ran for both population sizes, for each combination of δ and dependence degree. The combination of all three parameters means that the first set consists of 50 simulation runs. Simulation set 2: *popsize* is fixed to 10, whereas δ is in a finer grained segment $[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$. The segment for *depdeg* is the same as in the previous set. Simulation set 3: *popsize* is equal to 10, however in this case δ is dynamic, i.e. it changes during runtime. The initial values of δ are in the segment $[0.0, 0.3, 0.5, 0.7, 1.0]$. In the final set, two simulations were run, (i) only one agent has dynamic δ, (ii) all agents have dynamic δ. The segment for *depdeg* remains the same.

During all simulation runs, agents can decide to perform a task t_i, or can receive a request for that task. At the beginning of a task, the agent checks the list of dependencies. If there is any such dependency, then the agent chooses whom to ask for assistance, by consulting its list of known agents which are able to perform t_i. The selection is done in the following way. The agent perceived as the most helpful in the past is chosen with probability 0.7, or random with probability 0.3. The 0.3/0.7 ratio is arbitrary. This scheme helps the agents to explore their options. The perceived helpfulness (*ph*) of an agent is computed by dividing the times it has given a response over the total times it was requested for help (Eq. 3).

$$ph = \frac{Requests\ Handled}{Total\ Requests} \tag{3}$$

C1 Results. The simulations were conducted in order to verify the hypothesis that agents with dynamic willingness to give help complete more of their dependent tasks as compared to agents with static willingness. The corresponding results are visualized as heat maps (Fig. 6), in which the x-axis represents the *depdeg*, an the y-axis represents δ, and the intensity of the color represents the percentage of completed tasks summed over all agents in the population, computed as a mean over the three independent runs.

Outcomes from the first set of simulation runs are shown in Figs. 6a, b, d, and e. It is possible to observe that for low dependence degree (10%), agents with a low willingness to give help (0.0), complete roughly 30% of their dependent tasks. This figure agrees with the probability that an agent is able to achieve a task by itself, when asking for help has failed. On the contrary, agents with willingness to give help, complete more dependent tasks, without noticeably impacting DD. Moreover, *popsize* does not show to have an impact on neither CD (Figs. 6a, b) nor DD (Figs. 6c, d). This is the reason why *popsize* equal to 10 was used in the remaining simulations. The outcomes for the second set of simulation runs (given in c and f) are consistent with the first set.

The third set of simulation runs covers the case in which δ is dynamic. As such, in the y-axis for Figures g, j are shown δ's initial values, δ_{init}. It is observed that for lower dependence degree, the population as a whole accomplishes more dependent tasks, thus CD increases, as compared to the static δ scenarios. This holds for both cases, i.e. only one agent has dynamic δ (Figs. 6g, i) and all agents have dynamic δ (Figs. 6h, j). Furthermore, the maximum CD is reached in the case where all agents have dynamic δ. On the other hand, as the dependence degree increases, so does DD and this is consistent through the three sets of runs. When all tasks depend on some other task (i.e. dependence degree is equal to 100%), DD reaches its maximum value, and CD is circa 0.3, which represents the probability for an agent to accomplish the task by itself.

The parameter DD influences the willingness to give help, and as a result the behavior of the agent (as is observed in Figs. 6a, b). In these simulations, two thresholds $\theta_{low} = 0.3$ and $\theta_{high} = 0.7$, are used in order to regulate δ as a function of DD as follows. If DD is higher than θ_{high}, then δ will decrease with a step of $\delta_{step} = 0.05$. If DD is lower than θ_{low}, then δ will increase with the same δ_{step}. If DD lies between θ_{low} and θ_{high}, the agent will compare the current DD with the one before. If such difference is bigger than 0.01 in absolute value, then δ will be updated, i.e. will increase if DD has gone down, and decrease otherwise.

5.2 Setup and Results for C2

C2 Setup. Agents are implemented as ROS nodes [37]. The communication between them is realized by using (i) the publish/subscribe mechanism in ROS and (ii) a tailored action server mechanism able to handle multiple requests at once which extends ROS's Simple Action Server implementation. Agents continuously broadcast their identity and list of tasks they can perform to each other through the publish/subscribe. In order to issue requests for help (one-to-one communications), agents use the action-server mechanism.

Agent behavior was studied in three different simulation scenarios in order to verify the three hypotheses H1, H2, and H3. The first scenario (S1) addresses H1. The population of agents contains one individual with $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, while the rest is configured with $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$. Thus one agent has high willingness to give help and low willingness to ask for help, whereas the rest of the population has low willingness to give help and high willingness to ask

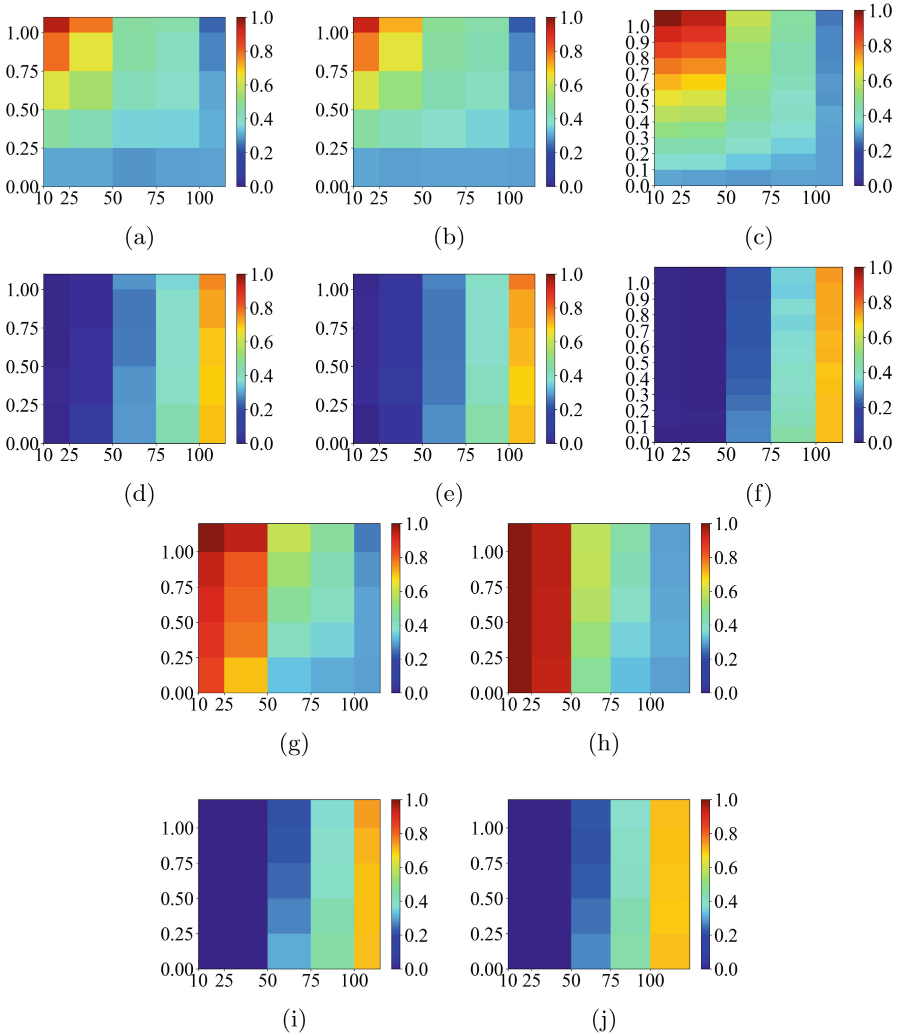


Fig. 6. Heat maps of CD and DD utility measures, for simulations with static δ and dynamic δ, and different popsize [7]. (a) CD for popsize = 10 with static δ. (b) CD for popsize = 30 with static δ. (c) CD for popsize = 10 with finer resolution of static δ. (d) DD for popsize = 10 with static δ. (e) DD for popsize = 30 with static δ. (f) DD for popsize = 10 with finer resolution of static δ. (g) CD for popsize = 10, one agent with dynamic δ. (h) CD for popsize = 10, all agents with dynamic δ. (i) DD for popsize = 10, one agent with dynamic δ. (j) DD for popsize = 10, all agents with dynamic δ.

for help. The values in the tuples change during the simulation. Two types of simulations are ran. (i) All agents calculate their δ based on Algorithm 2.a. The simulation time is equal to $t_s = 1\,\text{h}$. (ii) All agents calculate their δ by applying to Algorithm 2.a the extension provided in Algorithm 2.b. In this case simulation time is $t_s = 3\,\text{h}$. This is to ensure that enough interactions take place for the agents to be able to adapt.

The second scenario (S2) addresses H2. The opposite agent configuration is shown, i.e. one agent is configured with $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$, while the others start off with $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$. In this case, an agent with low willingness to give help and high willingness to ask for help is put among agents with high willingness to give help and low willingness to ask for help. Two types of simulations are ran. (i) All agents calculate their δ through Algorithm 2.a, and (ii) all agents calculate their δ by applying to Algorithm 2.a the extension provided in Algorithm 2.b.

In the final scenario (S3 which addresses H3), half of the population is configured with $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, while the other half is configured with $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$. As in the previous scenarios, two types of simulations are run. (i) All agents calculate their δ by using Algorithm 2.a. (ii) All agents calculate their δ by applying to Algorithm 2.a the extension provided in Algorithm 2.b.

Every simulation was repeated for a population size *popsize* equal to 10 and 30. In all cases the difficulty of the simulation, i.e. the probability that an agent needs the assistance of another agent for a task, is set to a low value equal to 0.2. The perceived willingness to help of an agent a_j is calculated through Eq. 3 by an agent a_i,

$$ph = \frac{rs}{rg} \tag{4}$$

where rs - total number of requests a_i has sent to a_j, rg - number of acceptance responses gotten from a_j. The agent to ask for help is chosen according to the following rules. As long as an agent a_i has a list of agents with which it does not have any past experience, it will choose randomly one of them. This is to ensure that an agent creates a past history with all the others early on in the simulation. Otherwise, the agent will choose randomly with a probability $P_1 = 0.4$ and according to Eq. 5 with probability $P_2 = 0.6$,

$$\beta = max(\{ph_1, ...ph_i, ...ph_n\}) \tag{5}$$

where ph_i - perceived helpfulness of agent i, n - number of agents. Agent risk in this paper is expressed as $1 - ph$. The culture variable a_k is calculated by averaging on the perceived willingness of all agents with which there is past experience, as given in Eq. 6,

$$C = \frac{\sum_{i=1}^{k} ph_i}{k} \tag{6}$$

where ph_i - perceived helpfulness of agent i, k - number of agents with which there is past experience. The agent's own performance is calculated as in Eq. 7.

$$\mu = \frac{tc}{ta} \tag{7}$$

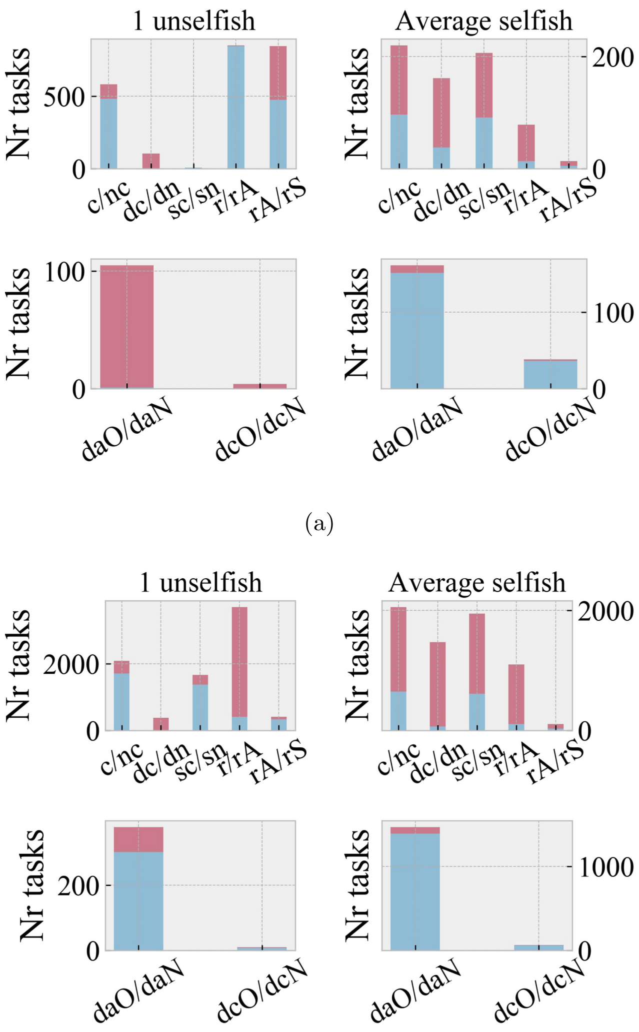
where tc - total tasks completed, ta - total tasks attempted. The variables for environmental risk and task progress do not change during the course of the simulation. Thus their effect on willingness does not change as well. Environmental risk is kept at a value equal to 0.2, and influences δ by $+\Delta step$, and γ by $-\Delta step$, where $\Delta step = 0.05$. Moreover, since reasoning on γ is done only at the beginning of a task, the agent cannot measure progress for itself, and thus the variable will affect γ by $-\Delta step$. Finally, since tasks are presently simulated as atomic steps, task trade-off has not been considered in these simulations.

C2 Results. The results shown in this subsection are taken as a mean over three independent runs, in which all parameter settings are the same, and visualized through bar plots. Moreover, for every simulation scenario, results corresponding to agents with the same initial configuration are averaged.

Figures 7 and 8 display results which address H1. Figure 7 shows the numerical values for a population of agents with $popsize = 10$, whilst Fig. 8 for $popsize = 30$. It is evident that the agent $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$ (Fig. 7, left side) adapts to the selfish population by comparing the rR/rRA ratio in Fig. 7(a) and (b). Note also that agents $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$ flood each other with the high number of requests made. Observing the bottom graphs in (a) and (b), it is possible to estimate how much an agent works for itself and for others by comparing daO/daNO ratios. Agent $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$ attempts dominantly more dependent tasks for others in (a), while the opposite is true in (b). This is reflected in the numbers for completed dependent tasks by comparing dco/dcNO ratios. Thus, H1 holds in the implemented scenario for both $popsize = 10,\ 30$.

Figures 9 and 10 show results which address H2. Figure 9 gives the values for a population of agents with $popsize = 10$, whilst 10 for $popsize = 30$. Note that, by comparing the dc/dnc ratio in Fig. 9(a) and (b), the selfish agent $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$ completes less dependent tasks in a society which will not be blindly helpful, but will tune helpfulness towards those individuals that respond in kind. As a result, the rest of the population ($\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$) will continue to do very well and will, after a period of learning, stop helping the selfish agent. Figure 9(c) considers a scenario in which the selfish agent $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$ adapts to the population of agents by becoming more helpful, thus it performs better than the agent in (b). To get the two different behaviors, the factor k in Algorithm 2.b was taken equal to 4 in (b), and 8 in (c). Figure 10 shows that the selfish agent completes a higher percentage of dependent tasks. The reason is argued to be the size of the population. It can take a population longer to adapt to a selfish individual because more of the required interactions need to take place. Until that point is reached, the selfish agent will do fairly well. Thus, H2 also holds in the implemented scenarios.

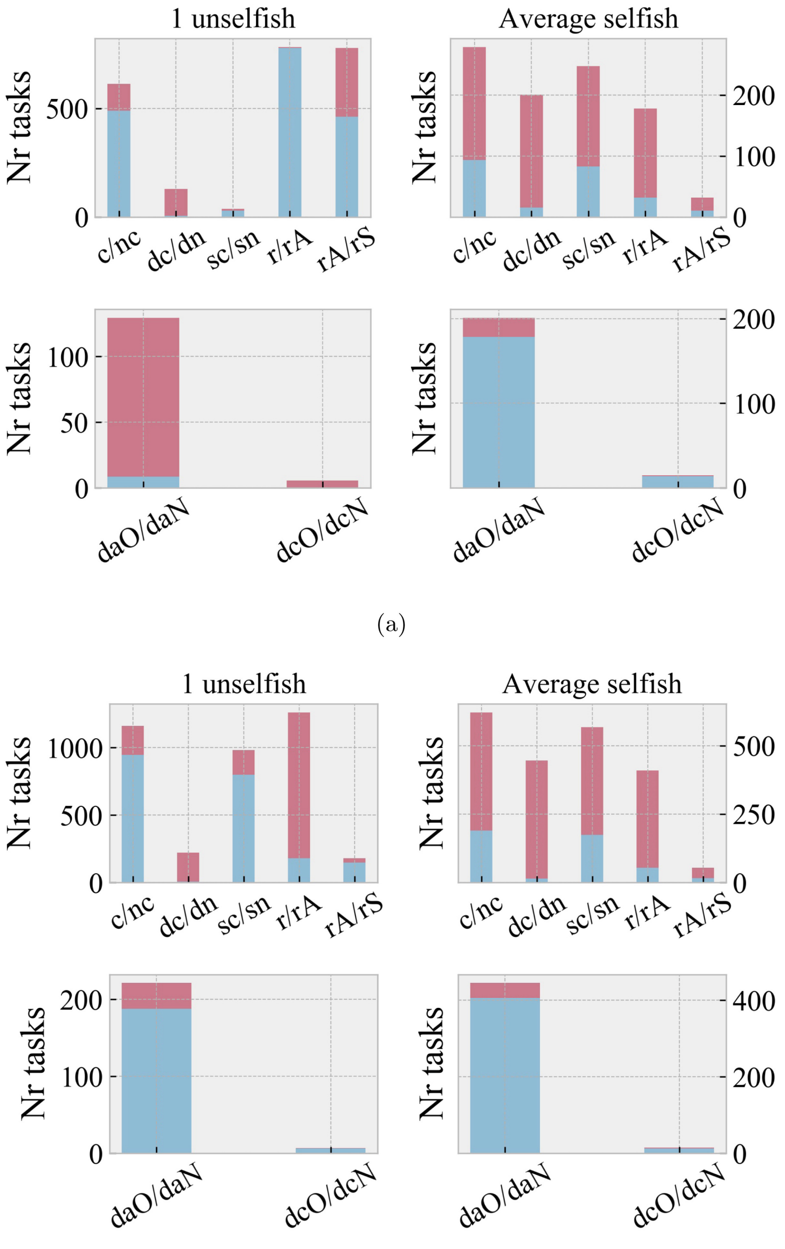
Figures 11 and 12 show results which address H3. Figure 11 gives the values for a population of agents with $popsize = 10$, whilst 12 for $popsize = 30$. Note that agents using Algorithm 2.b adapt to the part of the population configured to exploit. Moreover, a helpful agent will achieve more tasks in average than its opposite. This holds for both population sizes. Nevertheless, overall



(a)

(b)

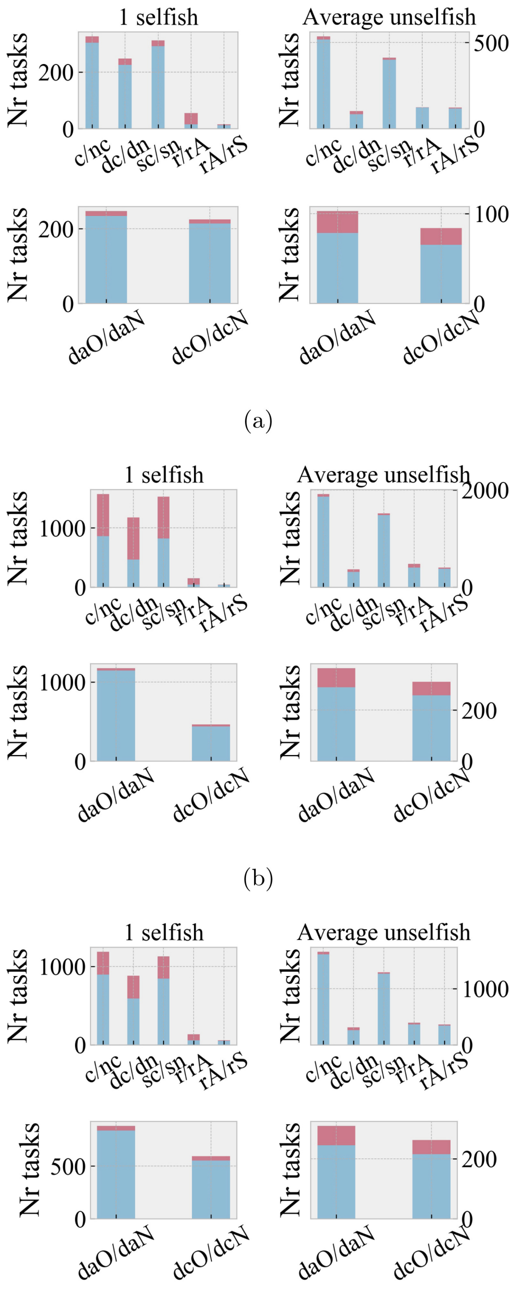
Fig. 7. Results for S1, for *popsize* = 10. Notation: c/nc - completed/not completed tasks (c - blue bottom bar, nc - top red bar), dc/dn - dependent completed/dependent not completed tasks, sc/sn - completed self-generated/not completed self generated, r/rA - request received/requests accepted, rA/rS - requests accepted/requests succeeded (as perceived by the agent who is helping, daO/daN - dependent self-generated tasks attempted/dependent not self-generated tasks attempted, dcO/dcN - dependent self-generated completed/dependent not self-generated completed). (a) Agents update δ with Algorithm 2.a. (b) Agents update δ with the extension proposed in Algorithm 2.b. Figures on the left correspond to the agent $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, while those on the right are averaged over the agents $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$ (Color figure online)



(a)

(b)

Fig. 8. Results for S1, for *popsize* $= 30$. Notation same as Fig. 7. (a) Agents update δ with Algorithm 2.a. (b) Agents update δ with the extension proposed in Algorithm 2.b. Figures on the left correspond to the agent $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, while those on the right are averaged over the agents $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$

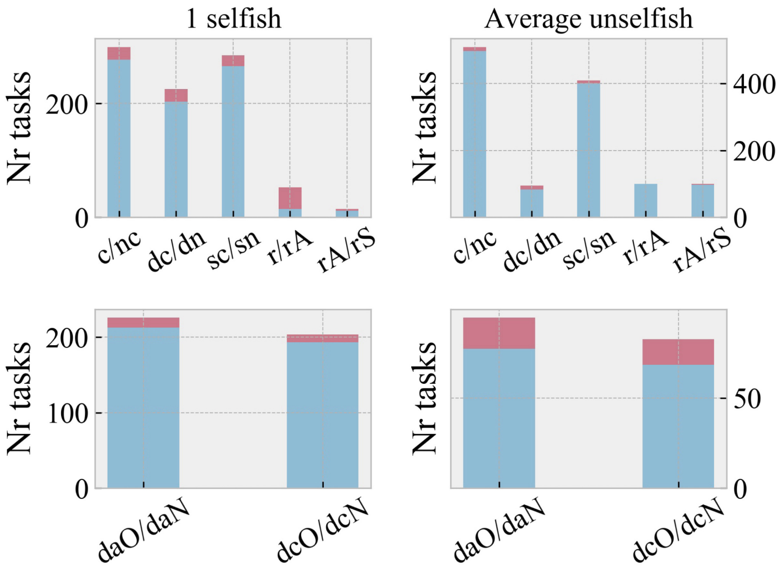


(a)

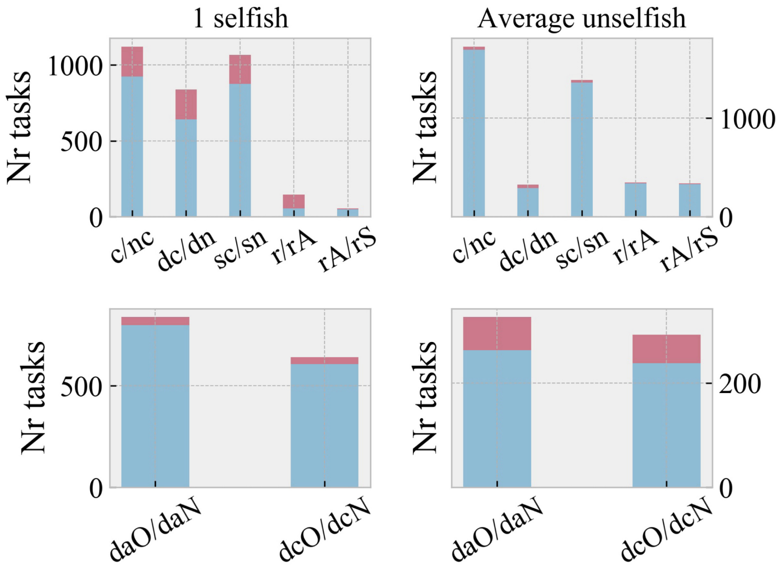
(b)

(c)

Fig. 9. Results for S2, for *popsize* = 10. Notation same as Fig. 7. (a) Agents update δ with Algorithm 2.a. (b) Agents update δ with the extension proposed in Algorithm 2.b. Figures on the left correspond to the agent $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$, while those on the right are averaged over the agents $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$. (c) ...



(a)



(b)

Fig. 10. Results for S2, for *popsize* $= 30$. Notation same as Fig. 7. (a) Agents update δ with Algorithm 2.a. (b) Agents update δ with the extension proposed in Algorithm 2.b. Figures on the left correspond to the agent $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$, while those on the right are averaged over the agents $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$

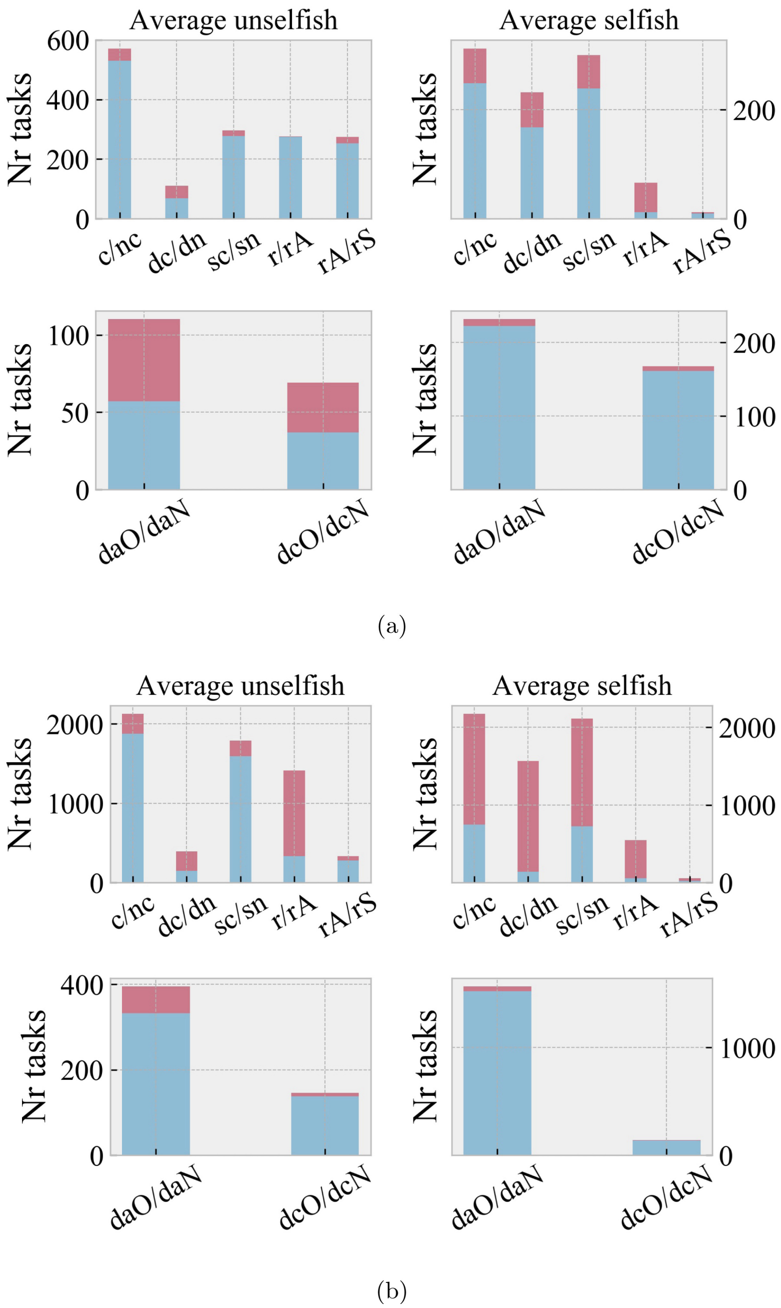
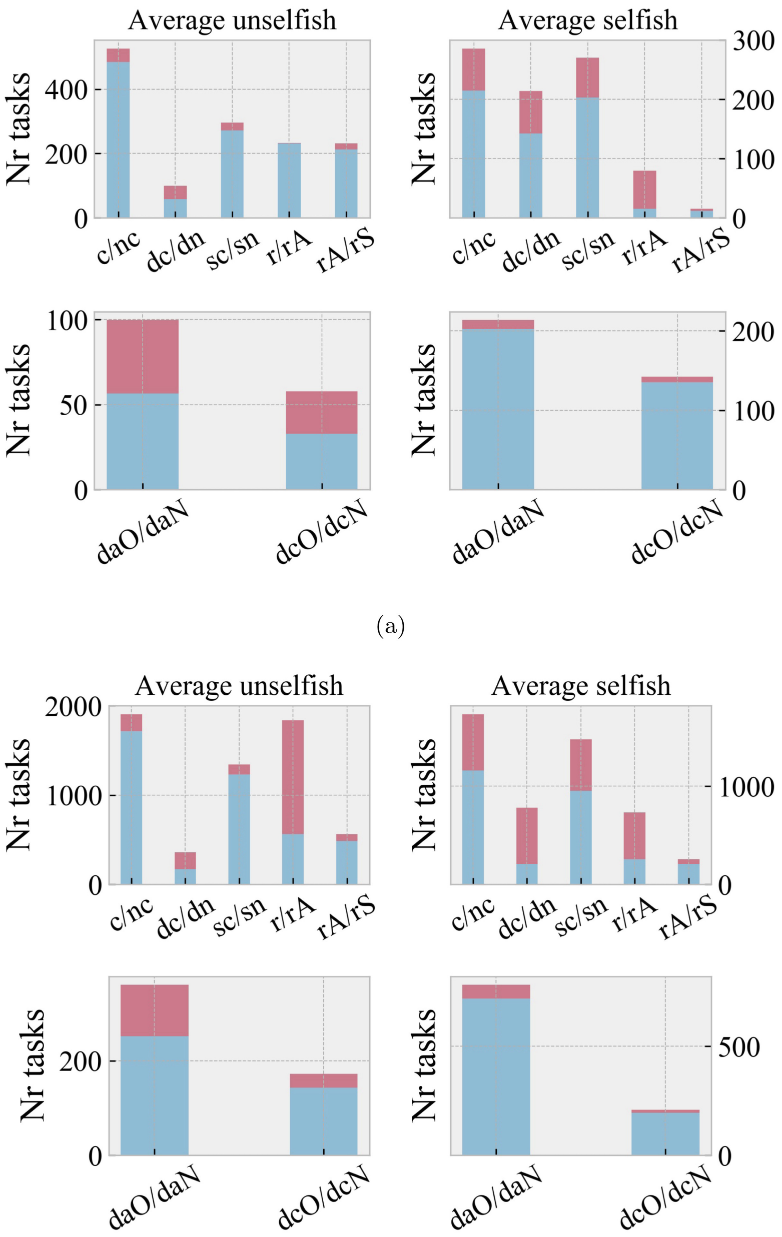


Fig. 11. Results for S3, for $popsize = 10$. Notation same as Fig. 7. (a) Agents update δ with Algorithm 2.a. (b) Agents update δ with the extension proposed in Algorithm 2.b. Figures on the left correspond to the average of half the population of agents with $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, while those on the right are averaged over the other half $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$



(a)

(b)

Fig. 12. Results for S3, for *popsize* = 30. Notation same as Fig. 7. (a) Agents update δ with Algorithm 2.a. (b) Agents update δ with the extension proposed in Algorithm 2.b. Figures on the left correspond to the average of half the population of agents with $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, while those on the right are averaged over the other half $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$

performance for the helpful agents falls (if compared with results in Figs. 9 and 10). This is because, the algorithm to choose an agent to ask for help does always pick from the entire population of agents, helpful and selfish combined. A finer selective method could be applied, so that helpful agents always pick between each other. The effects of the population size in the completed dependent tasks is mildly present, however it is not dominant. Thus, H3 also holds in the current implementation.

6 Discussion

The work presented in this paper describes an agent model that exhibits adaptive autonomous behavior by manipulating its willingness to interact, which in turn is composed by the willingness to give help (δ) and the willingness to ask for help (γ). The willingness to interact is expressed with probabilities in order to model the non-deterministic aspect of interactions. In some cases the agent might be completely sure that it needs help, e.g. lacks one of the abilities required for a task. However, in other cases this may not happen. For instance, assume an agent which is progressing slowly on a task. Its willingness to ask for help will increase. However, let us also assume that the agent is among rather unhelpful or unsuccessful agents. This element will decrease the willingness because the probability that those agent will be helpful now is low. In this scenario, the agent's willingness is influenced by contradicting factors that do not necessarily lead to a deterministic choice. There is a risk associated in choosing to ask or not ask for help. Consequently, the willingness to ask for help implicitly assesses this risk, and such assessment is probabilistic. In economics, this kind of parameter is used to model risk tolerance [38]. Agents which are representatives of business entities, sign contracts with other entities based on their willingness. Signing some contracts might be not allowed and thus subject to fines. The latter are considered punishment for undesired behavior. The higher the fines, the higher the risk is of signing a contract with an agent.

In the first part of this paper ($C1$), the agent consists only of δ, which is calculated based on the agent's own performance. In the second part ($C2$), implements the whole willingness to interact behavior. Additionally in $C2$, the perceived helpfulness of the population is used as an additional input in the calculation of δ. In this way, agents who are configured to be helpful can reduce exploitation by other agents which will not respond to helpfulness in kind. Algorithm 2.a was extended through Algorithm 2.b and used in the simulations, however other similar algorithms could be implemented that achieve the same end. Algorithm 2.b is purely experience based, and thus agents will need to experience a certain number of interaction before the learning takes place. Moreover, its efficiency – number of interactions needed for the learning – depends on the size of the population, as well as the ratios of selfish and unselfish agents. Furthermore, only two types of agents are used, $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$ and $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$. It can be argued that a mixed population will have additional effects on the results presented here.

This work also relates to computational trust models for multi-agent systems. The reason is that, the willingness to interact is used when deciding to depend upon another, and allowing others to depend on one as well. According to Falcone *et al.* [39], by trusting, an agent has made a decision to rely on another. The other way around, i.e. allowing others to depend on one, could also be argued as a case for trust. The agent that is helping, is putting its resources in the service of another. It is reasonable to think that help should go more towards those who have returned the favour. There is a fair amount of trust models in the literature, as well as classifications based on different criteria for them [40]. These models typically are experience-based (judging on personal direct interactions), reputation-based (judging based on third party opinions), or combined [40].

The aim of future work is to investigate the concept of *agent culture* and trust more thoroughly. Apart from perceived helpfulness, other characteristics could be used, for example perceived expertise, or perceived load. There is a difference between perceived helpfulness and perceived expertise. The latter is a better measure for successful outcomes of tasks. Assume an agent a_i which asks for help an agent a_j, and a_j accepts to help. However, being willing to help does not guarantee success, in fact agent a_j can fail for any reason. From this perspective, a flaw of perceived helpfulness as an indicator of helpfulness becomes apparent. As a result, a combined measure of perceived helpfulness and expertise can give a better picture of the agents taken as individuals and as a whole. Nevertheless, the degree of load of agents can help refine the measure even further. Consequently, the problem that arises is how can an agent estimate the load of other agents, taken individually and as a whole. The perceived load could be combined with the perceived willingness to ask for help from a specific agent and be used in its analysis, and that of the whole population. The implementation of the agent has to be expanded so that task execution is simulated through separate steps, so as to be interrupt-able, so that it represents a more realistic scenario. Another desired feature for the system is to enable the agent to deal with several dependencies at the same time.

Application domains that motivate this research include but are not limited to search and rescue, agriculture, and other areas in which autonomous systems can assist, and sometimes replace, human labour due to the challenging working conditions. It may be desired for these systems to be deployed far from the operator, and work in conditions where the communication with the former can be unreliable. Also, it may be desired that the systems operate by aiding each other as the needs arise. This means that agents need to reason about when and how to interact with each other. The AA agent described in this paper represents one possible approach that can be applied in these types of application. Moreover, the modeling is done on a high-level, and hence is not task specific in fact tasks are defined in abstract terms.

7 Conclusion

In this paper it is shown, through computer simulations, how the adaptive autonomous behavior of an agent can be tailored to reflect the culture of the population within which it operates, such that the exploitation of the agent can be decreased. Firstly, it is shown how an agent highly willing to give help, can adapt to agents configured oppositely, and thus lower its exploitation. Secondly, when an unhelpful agent is introduced into a helpful population, it will be able to exploit up until the point in which the rest of the agents will have created a history of negative past experiences. The dependence on the population size is also present in the simulation results, i.e. one selfish agent will be able to succeed for longer in bigger populations because more interactions are needed. Nevertheless, if the agent can change its behavior from selfish to unselfish, it will outperform agents which are more inflexible. Finally, when the population is split in half between selfish and unselfish agents, the latter is still able to decrease it exploitation after enough interactions have taken place. However, since the algorithm to select the agent is such that it will always pick from the whole pool, the performance of unselfish agents decreases as well, compared to the scenario in which the latter dominates. A finer selection algorithm could address this issue.

Appendix

The source code to replicate the C1 simulations is publicly available on github, under the following URL: https://github.com/gitting-around/gitagent_base.git. Whereas, for C2 simulations the source code is available under the following URL: https://github.com/gitting-around/gitagent.git. Finally, the simulations for this paper were conducted on HP EliteBook 840 laptop with Ubuntu 14.04 and ROS Indigo.

References

1. Hardin, B., Goodrich, M.A.: On using mixed-initiative control: a perspective for managing large-scale robotic teams. In: Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction, pp. 165–172. ACM (2009)
2. Johnson, M., Bradshaw, J.M., Feltovich, P.J., Jonker, C.M., van Riemsdijk, B., Sierhuis, M.: The fundamental principle of coactive design: interdependence must shape autonomy. In: De Vos, M., Fornara, N., Pitt, J.V., Vouros, G. (eds.) COIN -2010. LNCS (LNAI), vol. 6541, pp. 172–191. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21268-0_10
3. Fong, T., Thorpe, C., Baur, C.: Collaborative Control: A Robot-Centric Model for Vehicle Teleoperation, vol. 1. Carnegie Mellon University, The Robotics Institute, Pittsburgh (2001)
4. Brookshire, J., Singh, S., Simmons, R.: Preliminary results in sliding autonomy for assembly by coordinated teams. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2004. Proceedings, vol. 1, pp. 706–711. IEEE (2004)
5. Parasuraman, R., Sheridan, T.B., Wickens, C.D.: A model for types and levels of human interaction with automation. IEEE Trans. Syst. Man Cybernet. Part A Syst. Hum. **30**(3), 286–297 (2000)

6. Castelfranchi, C.: Founding agent's ‘autonomy’ on dependence theory. In: Proceedings of the 14th European Conference on Artificial Intelligence, pp. 353–357. IOS Press (2000)
7. Frasheri, M., Çürüklü, B., Ekström, M.: Towards collaborative adaptive autonomous agents. In: 9th International Conference on Agents and Artificial Intelligence 2017 ICAART, 24 February 2017, Porto, Portugal (2017)
8. Lewis, B., Tastan, B., Sukthankar, G.: An adjustable autonomy paradigm for adapting to expert-novice differences. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1656–1662. IEEE (2013)
9. Muszynski, S., Stückler, J., Behnke, S.: Adjustable autonomy for mobile teleoperation of personal service robots. In: RO-MAN, 2012 IEEE, pp. 933–940. IEEE (2012)
10. Birk, A., Pfingsthorn, M.: A HMI supporting adjustable autonomy of rescue robots. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 255–266. Springer, Heidelberg (2006). https://doi.org/10.1007/11780519_23
11. Goodrich, M.A., Olsen, D.R., Crandall, J.W., Palmer, T.J.: Experiments in adjustable autonomy. In: Proceedings of IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents, pp. 1624–1629 (2001)
12. Lin, L., Goodrich, M.A.: Sliding autonomy for UAV path-planning: adding new dimensions to autonomy management. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, pp. 1615–1624. International Foundation for Autonomous Agents and Multiagent Systems (2015)
13. Dorais, G., Bonasso, R.P., Kortenkamp, D., Pell, B., Schreckenghost, D.: Adjustable autonomy for human-centered autonomous systems. In: Working notes of the Sixteenth International Joint Conference on Artificial Intelligence Workshop on Adjustable Autonomy Systems, pp. 16–35 (1999)
14. Kortenkamp, D., Keirn-Schreckenghost, D., Bonasso, R.P.: Adjustable control autonomy for manned space flight. In: Aerospace Conference Proceedings, 2000 IEEE, vol. 7, pp. 629–640. IEEE (2000)
15. Calhoun, G.L., Goodrich, M.A., Dougherty, J.R., Adams, J.A.: Human-autonomy collaboration and coordination toward multi-RPA missions. In: Remotely Piloted Aircraft Systems: A Human Systems Integration Perspective, p. 109 (2016)
16. Barnes, M.J., Chen, J.Y., Jentsch, F.: Designing for mixed-initiative interactions between human and autonomous systems in complex environments. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1386–1390. IEEE (2015)
17. Stubbs, K., Hinds, P.J., Wettergreen, D.: Autonomy and common ground in human-robot interaction: a field study. IEEE Intell. Syst. **22**(2), 42–50 (2007)
18. van der Vecht, B., Dignum, F., Meyer, J.C.: Autonomy and coordination: controlling external influences on decision making. In: IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies, WI-IAT 2009, vol. 2, pp. 92–95. IEEE (2009)
19. Landén, D., Heintz, F., Doherty, P.: Complex task allocation in mixed-initiative delegation: a UAV case study. In: Desai, N., Liu, A., Winikoff, M. (eds.) PRIMA 2010. LNCS (LNAI), vol. 7057, pp. 288–303. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25920-3_20
20. Kim, S., Kim, M., Lee, J., Hwang, S., Chae, J., Park, B., Cho, H., Sim, J., Jung, J., Lee, H., et al.: Team SNU's control strategies for enhancing a robot's capability: lessons from the 2015 DARPA robotics challenge finals. J. Field Robot. **34**, 359–380 (2016)

21. Farinelli, A., Raeissi, M.M., Brooks, N., Scerri, P., et al.: Interacting with team oriented plans in multi-robot systems. Auton. Agents Multi-Agent Syst. **31**, 332–361 (2016)
22. Côté, N., Canu, A., Bouzid, M., Mouaddib, A.I.: Humans-robots sliding collaboration control in complex environments with adjustable autonomy. In: Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology, vol. 02, pp. 146–153. IEEE Computer Society (2012)
23. Chitalia, Y., Zhang, W., Hyun, B., Girard, A.: A revisit-based mixed-initiative nested classification scheme for unmanned aerial vehicles. In: American Control Conference, ACC 2014, pp. 1793–1798. IEEE (2014)
24. Feltovich, P.J., Bradshaw, J.M., Clancey, W.J., Johnson, M.: Toward an ontology of regulation: socially-based support for coordination in human and machine joint activity. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 175–192. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75524-1_10
25. Bradshaw, J.M., Jung, H., Kulkarni, S., Johnson, M., Feltovich, P., Allen, J., Bunch, L., Chambers, N., Galescu, L., Jeffers, R., et al.: Kaa: policy-based explorations of a richer model for adjustable autonomy. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, pp. 214–221. ACM (2005)
26. Scerri, P., Pynadath, D.V., Tambe, M.: Towards adjustable autonomy for the real world. J. Artif. Intell. Res. **17**(1), 171–228 (2002)
27. Goodrich, M.A., Schultz, A.C.: Human-robot interaction: a survey. Found. Trends Hum. Comput. Interact. **1**(3), 203–275 (2007)
28. Suzanne Barber, K., Goel, A., Martin, C.E.: Dynamic adaptive autonomy in multi-agent systems. J. Exp. Theor. Artif. Intell. **12**(2), 129–147 (2000)
29. Martin, C., Barber, K.S.: Adaptive decision-making frameworks for dynamic multi-agent organizational change. Auton. Agents Multi Agent Syst. **13**(3), 391–428 (2006)
30. Johnson, M., Bradshaw, J.M., Feltovich, P.J., Jonker, C.M., Van Riemsdijk, M.B., Sierhuis, M.: Coactive design: designing support for interdependence in joint activity. J. Hum. Robot Interact. **3**(1), 2014 (2014)
31. Klien, G., Woods, D.D., Bradshaw, J.M., Hoffman, R.R., Feltovich, P.J.: Ten challenges for making automation a "team player" in joint human-agent activity. IEEE Intell. Syst. **19**(6), 91–95 (2004)
32. Tambe, M.: Agent architectures for flexible. In: Proceedings of the 14th National Conference on AI, USA, pp. 22–28. AAAI press (1997)
33. Laird, J.E.: The Soar Cognitive Architecture. MIT Press, Cambridge (2012)
34. Schurr, N., Marecki, J., Tambe, M.: Improving adjustable autonomy strategies for time-critical domains. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 353–360. International Foundation for Autonomous Agents and Multiagent Systems (2009)
35. McGill, S.G., Yi, S.J., Yi, H., Ahn, M.S., Cho, S., Liu, K., Sun, D., Lee, B., Jeong, H., Huh, J., et al.: Team THOR's entry in the DARPA robotics challenge finals 2015. J. Field Robot. **34**, 775–801 (2016)
36. Castelfranchi, C., Falcone, R.: Principles of trust for MAS: cognitive anatomy, social importance, and quantification. In: International Conference on Multi Agent Systems, Proceedings, pp. 72–79. IEEE (1998)

37. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software, vol. 3, p. 5, Kobe (2009)
38. Cardoso, H.L., Oliveira, E.: Adaptive deterrence sanctions in a normative framework. In: IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies, WI-IAT 2009, vol. 2, pp. 36–43. IEEE (2009)
39. Falcone, R., Castelfranchi, C.: Social trust: a cognitive approach. In: Castelfranchi, C., Tan, Y.H. (eds.) Trust and Deception in Virtual Societies, pp. 55–90. Springer, Dordrecht (2001)
40. Pinyol, I., Sabater-Mir, J.: Computational trust and reputation models for open multi-agent systems: a review. Artif. Intell. Rev. **40**(1), 1–25 (2013)

Evaluating Task-Allocation Strategies for Emergency Repair in MAS

Hisashi Hayashi(✉)

System Engineering Laboratory, Corporate Research and Development Center,
Toshiba Corporation, 1 Komukai-Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan
`hayashi-hisashi@aiit.ac.jp`

Abstract. Nowadays, many systems are connected through networks. System of systems (SOSs) of this type can be regarded as multi-agent systems (MASs). These SoSs or MASs are robust against system failures because a failure of a system does not immediately mean the total failure of the whole system. In this paper, we consider a repairing problem of MASs where causes of future agent failures have to be removed within a limited time, and some agents become out of order if not repaired. In our simulation scenarios, many causes of future agent failures in MASs are found simultaneously and consecutively owing to large-scale disasters. In order to effectively repair them and reduce the number of agent failures, task-allocation strategies for emergency repair are extremely important. This paper compares five task-allocation algorithms in emergency situations: independent unit MAS algorithm, centralized algorithm, distributed algorithm, centralized algorithm with replanning, and distributed algorithm with replanning.

Keywords: Multi-agent systems · Coordination mechanism
Distributed task allocation · Emergency repair

1 Introduction

In general, multi-agent systems (MASs) are believed to be robust against partial system failures because even if an agent stops working, most of the other agents

---

This paper is modified and extended from our earlier conference paper [12] that was selected from ICAART2016 and ICAART2017 and recommended for submission to this journal. Compared with our previous work [12], we add two new sections: Sects. 2 and 7. We cite more related work in Sect. 2. We show new simulation results in Sect. 7 and modified the other sections accordingly. We also add Tables 5 and 6, Figs. 8, 9, 10 and 11. We reuse the other tables and figures although some of them are modified.

H. Hayashi—Presently with School of Industrial Technology, Advanced Institute of Industrial Technology, 1-10-40 Higashi-Ooi, Shinagawa-ku, Tokyo, 140-0011, Japan. (Since 1 October 2017).

This research was conducted at Toshiba Corporation.

© Springer International Publishing AG, part of Springer Nature 2018
N. T. Nguyen et al. (Eds.): TCCI XXVIII, LNCS 10780, pp. 253–274, 2018.
https://doi.org/10.1007/978-3-319-78301-7_11

continue to work and cover the task of the disabled agent unless some important agents break down. Usually, only a few causes of future agent failures are found simultaneously and it is not difficult to remove them before some agents actually fail. However, when large-scale disasters happen, many causes of future agent failures are created simultaneously and consecutively, in which case, the total MAS will stop functioning if they are not repaired effectively using limited resources within limited time. Therefore, it is vital to effectively allocate repair tasks.

In this paper, we consider some situations where disaster events repeatedly happen and each disaster event creates multiple causes of future agent failures In such emergency situations, we compare some task-allocation algorithms for MASs in order to reduce the number of actual agent failures. Although some task-allocation algorithms for repairing are compared in [3], the maximum number of causes of future agent failures is 10 in their test scenarios. On the other hand, in our much more severe scenarios, the maximum number of causes of future agent failures is of the order of hundreds. As discussed in Sect. 8, this difference leads to completely different conclusions. Although cooperation between agents was not effective in the test scenarios of [3], we predict that cooperation is effective in order to allocate limited resources within limited time when many causes of future agent failures are created simultaneously and consecutively owing to large-scale disasters.

As discussed in [8,17,22], there are two kinds of task-allocation algorithms: centralized algorithms and distributed algorithms. In centralized algorithms, a single manager agent collects information from its child agents, calculates the combination of tasks and child agents, and allocates the tasks to them. On the other hand, in distributed algorithms, multiple manager agents communicate with one another in order to decide on task allocations. Many existing task-allocation algorithms are centralized algorithms. However, distributed task-allocation algorithms are attracting attention because the breakdown of one manager agent does not lead to the total failure of the MAS. One of our aims is to compare these two kinds of algorithms in the repair-task-allocation scenarios. Because repair actions sometimes fail, it is natural to replan and reallocate the repair task in the case of an action failure. Therefore, we also evaluate centralized and distributed task-allocation algorithms with replanning capabilities. In summary, we compare five task-allocation algorithms for emergency repair: independent unit MAS algorithm (baseline algorithm where manager agents do not cooperate with one another), centralized algorithm, distributed algorithm, centralized algorithm with replanning, and distributed algorithm with replanning.

The remainder of this paper is organized as follows. In Sect. 2, we discuss related work. In Sect. 3, we define the MAS architecture for repair-task allocations. In Sect. 4, we define five algorithms for task repairing. In Sect. 5, we explain the detailed settings for simulation. In Sect. 6, we show the simulation results for a deployment pattern of unit MASs. In Sect. 7, we show the simulation results for another deployment pattern of unit MASs. In Sect. 8, we discuss the simulation results, comparing them with the simulation results reported in [3]. Section 9 is devoted to the conclusion.

2 Related Work

This section summarizes related work on task allocation. Auction algorithms such as the contract net protocol [26] are often used for dynamic task allocation [1,3,7,10,16]. As shown in this paper, it is not difficult to use auction algorithms in a decentralized manner.

In [22], the probabilities of future agent failures are taken into consideration when allocating tasks to agents. However, the algorithm does not consider repairing. In [11], some backup agents are created in case of emergency. However, as pointed out in [22], the cost of backup agents is high when additional hardware is needed and it takes time to copy the agents dynamically. The algorithm does not consider repairing either. In [20], considering future agents' failure, robust agent teams are created. The idea is to prepare more agents than needed. Again, the cost is high and repairing is not considered. Coalition formation of first responders in disaster relief is also researched in [23].

Metaheuristics are often used for optimizing combination of tasks and agents considering various constraints. In [15,31], multiple metaheuristics for task allocation are compared. In [31], it is shown that a variant of tabu search is better than other algorithms in terms of computation time and optimality. In [15], it is shown that a variant of PSO produces slightly better results in terms of optimality when the computation time is limited. However, in general, the algorithms of metaheuristics are time-consuming.

Variants of max-sum algorithms for distributed constraint optimization (DCOP) are applied to task-allocation problems in [17,23]. Most algorithms of DCOP use connectivity graphs of agents. In the case of agent failures, these DCOP algorithms do not update connectivity graphs of agents. On the other hand, max-sum is robust against agent failures. However, many messages are repeatedly sent between agents in DCOP, which causes delays of communication and computation.

There are many applications of multi-agent task allocation: disaster relief [1,6,19,21,23–25,28], computer games [9,27], coordination of robots [10,16,18, 30], weapon-target assignment [7,15,31], and command and control for combat ships [2–5,33]. Our task-allocation problem of emergency repair is closely related to the task-allocation problems of combat ships and disaster relief. Similar to our task-allocation problem, there are hard deadlines for tasks such as threat removal and civilian rescue.

3 MAS Architecture and Problem Description

We consider a MAS for repair-task allocations that is composed of multiple **unit MAS**s, each of which includes **sensing agents**, **action-execution agents**, and a **manager agent**: sensing agents detect causes of future agent failures, action-execution agents fix causes of future agent failures using limited resources, and manager agents communicate with one another to allocate repair tasks to action-execution agents. In this section, we define unit MASs and the agents that

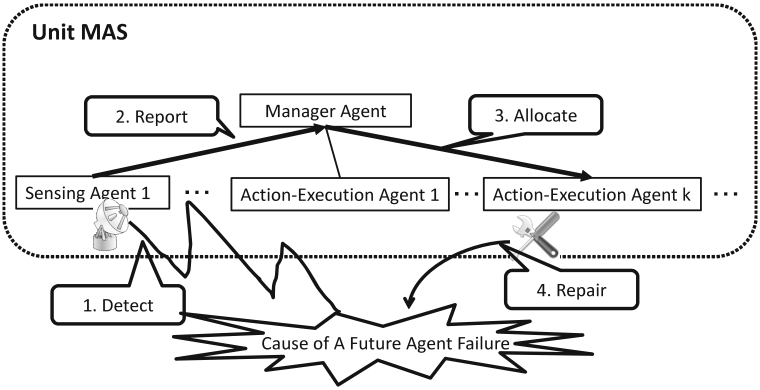


Fig. 1. MAS architecture for repair-task allocations

belong to unit MASs. We define the functions of unit MASs as agents because each function is often deployed on different hardware and becomes out of order independently.

As shown in Fig. 1, a unit MAS is a MAS comprising 0 or more sensing agents, 0 or more action-execution agents, and 1 manager agent. When a sensing agent senses a cause of a future agent failure, it reports the information to the manager agent in the same unit MAS. When receiving the information of a cause of a future agent failure, the manager agent allocates the repair task to an action-execution agent that belongs to the same unit MAS or allocates the repair task to the manager agent of another unit MAS if there are multiple unit MASs and their manager agents are connected by the network.

When allocated a repair task, the action-execution agent will execute a repair action consuming one resource. Execution of a repair action will succeed or fail according to the predefined probability. Unless a cause of a future agent failure is removed by a repair action, one of the agents will stop functioning according to the predefined probability.

When a unit MAS has sensing agents and action-execution agents, the unit MAS can sense and remove causes of future agent failures without the help of the other unit MASs as shown in Fig. 1. In this case, each unit MAS is **independent** and it is unnecessary to connect each unit MAS through the network. Because of its simplicity, we compare the repair-task-allocation algorithm for independent unit MASs with other algorithms as a baseline algorithm.

As shown in Fig. 2, when the manager agents of unit MASs are connected by the network, sensing and repairing can be done in different unit MASs. We expect that repair-task allocations will be more effective when the manager agents are connected.

When the MAS has a **centralized** architecture, as shown in Fig. 3, only one manager agent (top manager agent) works as a leader and allocates all the repair tasks to unit MASs. On the other hand, when the MAS has a **distributed** architecture as shown in Fig. 2, different manager agents become leaders of task

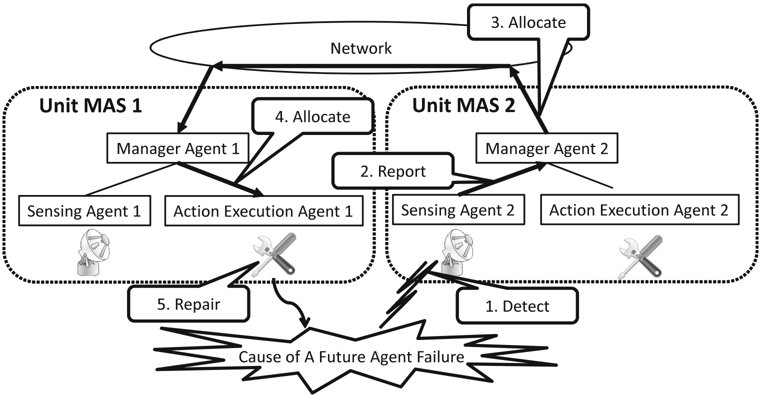


Fig. 2. Distributed architecture

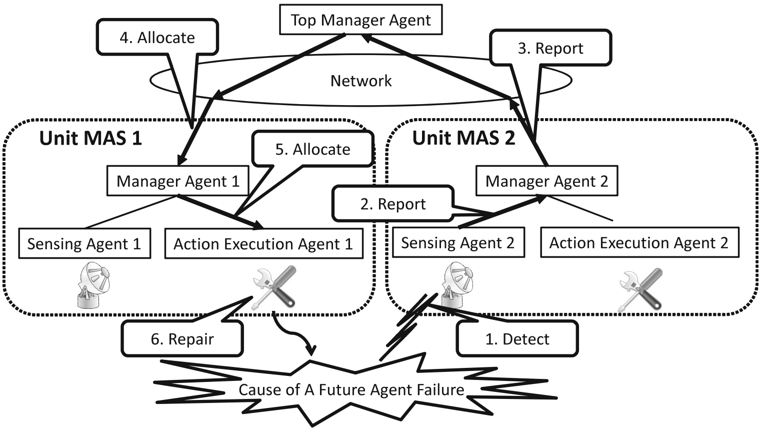


Fig. 3. Centralized architecture

allocation for different repair tasks. We expect that the distributed MAS algorithm is more robust than the centralized algorithm because in the distributed architecture, even if a leader becomes out of order, another manager agent becomes a leader. We compare the repair-task-allocation algorithms that use these MAS architectures.

4 Algorithms

In this section, we introduce five algorithms for repair-task allocations: the independent unit MAS algorithm (baseline algorithm), the centralized algorithm, the distributed algorithm, the centralized algorithm with replanning, and the distributed algorithm with replanning. Repair tasks are allocated to agents based on the contract net protocol [26].

In the independent unit MAS algorithm, the manager agent of each unit MAS decides which repair action to execute without exchanging information with other unit MASs. In the centralized algorithm, only one manager agent (top manager agent) works as a leader and allocates repair tasks to unit MASs. In the distributed algorithm, when a sensing agent detects a cause of a future agent failure, the manager agent in the same unit MAS works as a leader and selects a unit MAS for the repair task. Note that different manager agents become leaders for different repair tasks in the distributed algorithm. In the centralized algorithm with replanning, only the top manager agent reallocates unsuccessful repair tasks to unit MASs. On the other hand, in the distributed algorithm with replanning, when an action-execution agent fails to execute a repair action, the manager agent in the same unit MAS works as a leader and reallocates the unsuccessful repair task to a unit MAS.

We expect that the centralized algorithm and the distributed algorithm can allocate repair tasks more effectively than the independent unit MAS algorithm because unit MASs communicate with one another. We also expect that the distributed algorithm is more robust than the centralized algorithm that is weak with respect to the failure of the top manager agent. We also evaluate the centralized algorithm with replanning and the distributed algorithm with replanning because it is natural and effective to reallocate the repair task when the repair task is not completed successfully.

4.1 Independent Unit MAS Algorithm

In the independent unit MAS algorithm, the manager agents of different unit MASs do not communicate with one another. Therefore, as in Fig. 1, when a sensing agent senses a cause of a future agent failure, the action-execution agent in the same unit MAS tries to repair it without the help of the other unit MASs. Although this algorithm does not require network connections among unit MASs, multiple action-execution agents in different unit MASs might try to execute repair actions for the same cause of a future agent failure, which leads to unnecessary consumption of resources.

Algorithm 1 (Independent Unit MAS Algorithm). *The sensing agents, the manager agent and the action-execution agents in each unit MAS work as follows if they are alive:*

- *Algorithm of Sensing Agents*
  1. *When a sensing agent detects a new cause of a future agent failure, it reports the information to the manager agent in the same unit MAS if the manager agent is alive.*
- *Algorithm of Manager Agents*
  1. *When the manager agent M receives the information of a new cause of a future agent failure C from a sensing agent in the same unit MAS, the manager agent M selects and reserves an action-execution agent E for the repair task R of C, if E exists, such that E is alive, the number of resources of E is more than 0 and E is not reserved for another cause of a future agent failure.*

  2. *When it becomes possible for the reserved action-execution agent E to start executing the repair action A for the reserved repair task R, if E is alive, the manager agent M in the same unit MAS orders E to execute A and erases the reservation information.*
- *Algorithm of Action-Execution Agents*
  1. *When receiving an execution order of the repair action A, from the manager agent M in the same unit MAS, the action-execution agent E executes A, decrements 1 resource whether the result of A is a success or a failure, and reports the result to M.*

4.2 Centralized Algorithm

A centralized algorithm is often used for task allocations in general. In the centralized algorithm, as shown in Fig. 3, only one manager agent (top manager agent) allocates repair tasks to the manager agents of unit MASs based on the contract net protocol [26], which is a kind of auction. Although there are many criteria to select a unit MAS, we select the unit MAS that can start repairing first. Because of the network connection, we expect that the centralized algorithm is more effective than the independent unit MAS algorithm.

Algorithm 2 (Centralized Algorithm). *The sensing agents, the manager agent and the action-execution agents in each unit MAS work as follows if they are alive:*

- *Algorithm of Sensing Agents*
  - *Same as the algorithm of sensing agents in Algorithm 1.*
- *Algorithm of Manager Agents*
  1. *When the manager agent M receives the information of a new cause of a future agent failure C from a sensing agent in the same unit MAS, the manager agent M forwards the information of C to the top manager agent T if T is alive. Otherwise, it works in the same way as step 1 of the algorithm of manager agents in Algorithm 1.*
  2. *When the manager agent M receives an allocation of a repair task R from the top manager agent, M selects and reserves an action-execution agent E for R in the same way as step 1 of the algorithm of manager agents in Algorithm 1.*
  3. *Same as step 2 of the algorithm of manager agents in Algorithm 1.*
- *Algorithm of the Top Manager Agents*
  1. *When the top manager agent T receives the information of a new*[1] *cause of a future agent failure C from a manager agent, T asks each alive manager agent M whether the unit MAS U of M can be in charge of the repair task R of C and how quickly an action-execution agent of U can start the repair action of R. Then, T allocates R to the manager agent of the unit MAS U2 such that an action-execution agent of U2 can start the repair action of R the quickest.*
- *Algorithm of Action-Execution Agents*
  - *Same as the algorithm of action-execution agents in Algorithm 1.*

---

[1] Even if multiple manager agents report the same cause of a future agent failure, the top manager agent starts the auction only once.

4.3 Distributed Algorithm

In the distributed algorithm, as shown in Fig. 2, when a sensing agent detects a cause of a future agent failure, the manager agent in the same unit MAS works as a leader like the top manager agent of the centralized algorithm, selects a unit MAS based on the contract net protocol [26], and allocates the repair-task to the manager agent of the selected unit MAS. However, unlike the centralized algorithm, different manager agents become leaders for different repair tasks. Therefore, the distributed algorithm is expected to be more robust for agent failures than the centralized algorithm.

Algorithm 3 (Distributed Algorithm). *The sensing agents, the manager agent and the action-execution agents in each unit MAS work as follows if they are alive:*

- *Algorithm of Sensing Agents*
  - *Same as the algorithm of sensing agents in Algorithm 1.*
- *Algorithm of Manager Agents*
  1. *When the manager agent M receives the information of a new*[2] *cause of a future agent failure C from a sensing agent in the same unit MAS, M asks each alive manager agent, selects a unit MAS U and allocates the repair task of C to the manager agent of the unit MAS U in the same way as step 1 of the algorithm of the top manager agent in Algorithm 2.*
  2. *When the manager agent M receives an allocation of a repair task R from the manager agent of another unit MAS, M selects and reserves an action-execution agent E for R in the same way as step 1 of the algorithm of manager agents in Algorithm 1.*
  3. *Same as step 2 of the algorithm of manager agents in Algorithm 1.*
- *Algorithm of Action-Execution Agents*
  - *Same as the algorithm of action-execution agents in Algorithm 1.*

4.4 Centralized Algorithm with Replanning

In the centralized algorithm with replanning, when execution of a repair action results in failure, the top manager agent reselects a unit MAS and reallocates the repair task to the manager agent of the selected unit MAS. Because the success rate of repair actions is not 100%, it is expected that replanning will decrease the number of agent failures.

---

[2] Even if a sensing agent reports a cause of a future agent failure to its manager agent, the manager agent will not start the auction if another manager agent has already started the auction for the same cause of a future agent failure. In this paper, for simplicity, we assume that there is no delay for agent communication. However, when there is such communication delay, each manager agent needs to stop the auction it has already started when they know that another manager agent has started the auction earlier for the same cause of a future agent failure. We are currently tackling this problem and recently reported some results in [13].

Algorithm 4 (Centralized Algorithm with Replanning). *The sensing agents, the manager agent and the action-execution agents in each unit MAS work as follows if they are alive:*

- *Algorithm of Sensing Agents*
  - *Same as the algorithm of sensing agents in Algorithm 1.*
- *Algorithm of Manager Agents*
  1. *Same as step 1 of the algorithm of manager agents in Algorithm 2.*
  2. *Same as step 2 of the algorithm of manager agents in Algorithm 2.*
  3. *Same as step 3 of the algorithm of manager agents in Algorithm 2.*
  4. *When the manager agent M receives the result of repair-action execution for the repair task R from an action-execution agent in the same unit MAS, if the result is a failure, M reports the result of R as a failure to the top manager agent.*
- *Algorithm of the Top Manager Agents*
  1. *Same as step 1 of the algorithm of the top manager agent in Algorithm 2.*
  2. *When the top manager agent T receives the result of a failure for the repair task R from a manager agent, T asks each alive manager agent, selects a unit MAS U, and reallocates R to one of the manager agents in the same way as step 1 of the algorithm of the top manager agent in Algorithm 2.*
- *Algorithm of Action-Execution Agents*
  - *Same as the algorithm of action-execution agents in Algorithm 1.*

Algorithm 5 (Distributed Algorithm with Replanning). *In the distributed algorithm with replanning, when an action-execution agent fails to execute a repair action, the manager agent in the same unit MAS reselects a unit MAS and reallocates the task to the manager agent of the selected unit MAS. It is expected that replanning will decrease the number of agent failures. The sensing agents, the manager agent and the action-execution agents in each unit MAS work as follows if they are alive:*

- *Algorithm of Sensing Agents*
  - *Same as the algorithm of sensing agents in Algorithm 1.*
- *Algorithm of Manager Agents*
  1. *Same as step 1 of the algorithm of manager agents in Algorithm 3.*
  2. *Same as step 2 of the algorithm of manager agents in Algorithm 3.*
  3. *Same as step 3 of the algorithm of manager agents in Algorithm 3.*
  4. *When the manager agent M receives the result of repair-action execution for the repair task R from an action-execution agent in the same unit MAS, if the result is a failure, this manager agent M reallocates R to one of the manager agents in the same way as step 1 of the algorithm of manager agents in Algorithm 3.*
- *Algorithm of Action-Execution Agents*
  - *Same as the algorithm of action-execution agents in Algorithm 1.*

5 Simulation Settings

In this section, we explain the details of simulation settings used to compare and evaluate the algorithms defined in the previous section. We evaluate the algorithms by means of simulation because the situation continuously changes and each cause of a future agent failure is detected asynchronously. The target application of our empirical case study is related to the application of [3] (weapon-target assignment), and we will compare our simulation results with the results in [3] later in the discussion section. We set typical values of unit MASs, considering the target application. In Sect. 5.1, we show the number of agents and resources in unit MASs. In Sect. 5.2, we show the performances of sensing and repairing of each unit MAS. In Sect. 5.3, we show the consecutive occurrence patterns of disaster events that create multiple causes of future agent failures simultaneously. As stressed in the introductory section, we predict that cooperation among agents becomes very effective in these severe occurrence patterns of disaster events, which is completely different from the result in [3] where the numbers of causes of future agent failures in their test scenarios are small.

5.1 The Numbers of Agents and Resources

This subsection shows the numbers of agents and resources. As shown in Table 1, we use 7 kinds of unit MASs: UMAS 0, ..., UMAS 6, which are typical unit MASs of our target application.

The numbers of action-execution agents in UMAS 0, ..., UMAS 6 are 0, 0, 0, 4, 2, 1, 1. An action-execution agent cannot execute more than one repair action in parallel but multiple action-execution agents can execute repair actions at the same time. The numbers of initial resources that each action-execution agent in UMAS 3, ..., UMAS 6 has are 18, 12, 24, 3.

UMAS 1 and UMAS 2 do not have action-execution agents, which means that the causes of agent failures found by the sensing agent of UMAS 1 or UMAS 2 need to be repaired by the action execution agents of UMAS 3, ..., UMAS 6. UMAS 0 does not have sensing agents or action-execution agents. It has only one

Table 1. The numbers of agents and resources

Type of unit MAS	# of manager agents	# of sensing agents	# of action-execution agents	# of initial resources of each action-execution agent
UMAS 0	1	0	0	-
UMAS 1	1	1	0	-
UMAS 2	1	1	0	-
UMAS 3	1	1	4	18
UMAS 4	1	1	2	12
UMAS 5	1	1	1	24
UMAS 6	1	1	1	3

Table 2. Probability and time to start detecting a cause before an agent failure

Type of unit MAS	Performance of sensing		Performance of repairing		
	Time (secs) to start detecting a cause before an agent failure	Prob. of detecting causes of future agent failures (%)	Time (secs) to start repairing before an agent failure	Time for repairing (secs) when starting the Repair x secs before an agent failure	Success Prob. of repairs (%)
UMAS 1	360	90	-	-	-
UMAS 2	180	90	-	-	-
UMAS 3	72	90	36	x/2.5	80
UMAS 4	43.2	90	18	x/2.5	80
UMAS 5	18	90	10.8	x/2.5	80
UMAS 6	18	90	10.8	x/2.5	80

Table 3. Occurrence patterns of disasters and causes of future agent failures

# of disaster events	# of causes of future agent failures created by a disaster event	Interval between disaster events (hours)	Interval of causes of future agent failures (secs)	Times from occurrence of a cause to an agent failure (secs)	Prob. of agent failures when not repaired (%)
10	30	1	1	1800	90

manager agent that works as the top manager agent when using the centralized algorithm or the centralized algorithm with replanning.

5.2 Performances of Sensing Agents and Action-Execution Agents

This subsection shows performances of unit MASs in terms of sensing agents and action-execution agents. With a specific application in mind, we set the typical performances of each unit MAS. Table 2 shows performances of unit MASs in terms of sensing and repairing. Sensing agents in UMAS 1, ..., UMAS 6 can start detecting causes of agent failure respectively from 360, 180, 72, 43.2, 18, 18 s before the expected time of agent breakdown. The sooner the sensing agent detects a cause of a future agent failure, the higher the performance is, which means that performance of the sensing agent in UMAS 1 is the best. The probability of detecting causes of future agent failures is 90%.

Action-execution agents in UMAS 3, ..., UMAS 6 can start repairing from 36, 18, 10.8, 10.8 s before the expected time of an agent failure. The sooner the action-execution agent can start repairing, the higher the performance is, which means that performance of the action-execution agent in UMAS 3 is the best. The success probability of repairing is 80%. When an action-execution agent starts repairing x seconds before the expected time of agent breakdown, the time of repairing will be x/2.5 s. We assume a situation where a cause of agent failure approaches the target agent at constant speed and the action-execution agent sends the resource for a repair to the cause of future agent failure at constant speed.

Table 4. Selection rules of agents

Random selection rule of agents	Concentrated selection rule of agents
One agent is randomly selected	The top manager agent is selected with the prob. of 10% and one agent is randomly selected otherwise

5.3 Occurrence Patterns of Disasters and Agent Failures

Table 3 summarizes the occurrence patterns of disaster events and causes of future agent failures. In our simulation scenarios, when a disaster event occurs, a cause of a future agent failure is created every second. The total number of causes of future agent failures created by a disaster event is 30. Disaster events repeatedly happen 10 times, and the interval between disaster events is 1 h. Note that the number of action-execution agents is 32, which is also the maximum number of the repair actions that can be executed in parallel. This number is closer to the number of causes of future agent failures created by a disaster event. Therefore, the repairing capability of the MAS is close to the limitation.

As summarized in Table 4, we use two different selection rules of agents: the random selection rule of agents and the concentrated selection rule of agents. When the random selection rule of agents is applied, if a cause of a future agent failure is not removed by a repair action, one of the agents is randomly selected. The selected agent becomes out of order with the probability of 90% in 1800 s. When the concentrated selection rule of agents is applied, the top manager agent (the manager agent of UMAS 0) is selected with the probability of 10% and one of the other agents is randomly selected if the top agent is not selected. We expect that the concentrated selection rule affects the centralized algorithm. When the concentrated selection rule is applied, the top manager agent is more likely to fail. Note that after the failure of the top manager agent, the centralized algorithm works in the same way as the independent unit MAS algorithm and each unit MAS does not communicate with one another.

6 Simulation 1 (Balanced deployment of unit MASs)

This section shows the simulation results for a balanced deployment pattern of unit MASs. (In the next section, unit MASs are deployed in a different pattern.) We conduct the simulation based on the simulation settings defined in the previous section using the algorithms defined in Sect. 4. Table 5 shows the number of each unit MAS that is used for the simulation in this section. (Note that different

Table 5. The number of unit MASs (Balanced deployment of unit MASs)

Type of unit MAS	UMAS 0	UMAS 1	UMAS 2	UMAS 3	UMAS 4	UMAS 5	UMAS 6
# of unit MASs	1	1	2	2	4	8	8

numbers of unit MASs are deployed in the simulation in the next section.) High-performance unit MASs are costly in general. Therefore, in the simulation of this section, considering the balance, we use a smaller number of high-performance unit MASs and a larger number of low-performance unit MASs.

The numbers of UMAS 0, ..., UMAS 6 are 1, 1, 2, 2, 4, 8, 8. The total number of these unit MASs is 26 (=1 + 1 + 2 + 2 + 4 + 8 + 8). Note that from Tables 2 and 5, we can see that the lower the performance of sensing and repairing of a unit MAS, the greater the number of the unit MAS.

UMAS 0 is necessary to evaluate the centralized algorithms. Each UMAS has exactly one manager agent and the total number of manager agents is 26. Each UMAS has exactly one sensing agent except UMAS 0 and the total number of sensing agents is 25. The numbers of action-execution agents in UMAS 0, ..., UMAS 6 are 0, 0, 0, 4, 2, 1, 1 (See Table 1). The total number of action-execution agents is 32 (=2 * 4 + 4 * 2 + 8 * 1 + 8 * 1). The numbers of initial resources that each action-execution agent in UMAS 3, ..., UMAS 6 has are 18, 12, 24, 3 (See Table 1). The total number of initial resources is 456 (=2 * 4 * 18 + 4 * 2 * 12 + 8 * 1 * 24 + 8 * 1 * 3).

We conducted simulations 1000 times using different random seeds. We show the results in terms of the number of agent failures and successful repairs. We show the simulation results in the following two subsections. In Sect. 6.1, we show the simulation results when using the random selection rule of agents. In Sect. 6.2, we show the simulation results when using the concentrated selection rule of agents.

6.1 When Using the Random Selection Rule of Agents

This subsection shows the simulation results when using the random selection rule of agents. Figure 4 shows the average number of agent failures and Fig. 5 shows the average number of successful repairs. Reducing agent failures is the top priority but it depends on successful repairs.

The centralized algorithm and the distributed algorithm are much better than the independent unit MAS algorithm because each cause of future agent failure

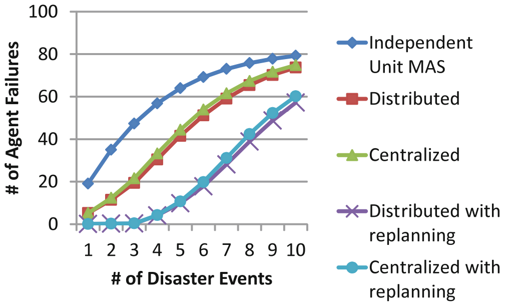


Fig. 4. Average number of agent failures when using the random selection rule of agents (Balanced deployment of unit MASs)

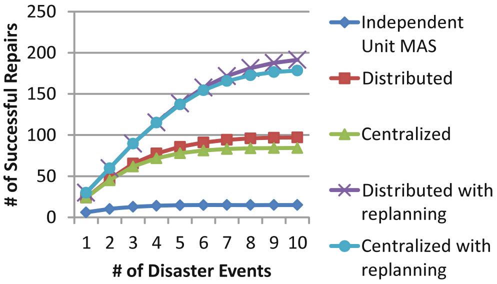


Fig. 5. Average number of successful repairs when using the random selection rule of agents (Balanced deployment of unit MASs)

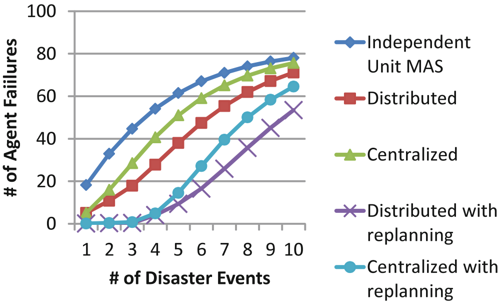


Fig. 6. Average number of agent failures when using the concentrated selection rule of agents (Balanced deployment of unit MASs)

is allocated to a unit MAS in the centralized algorithm and the distributed algorithm, which is not the case with the independent unit MAS algorithm. These two algorithms become much better when combined with a replanning capability because replanning covers unsuccessful repair actions.

The distributed algorithm with/without replanning is better than the centralized algorithm with/without replanning. However, the difference is slight. We expect that the difference would become bigger after the top manager agent becomes out of order. Therefore, in the next subsection, we focus on use of the concentrated selection rule of agents to increase the possibility of a failure of the top manager agent.

In summary, in this simulation scenario, the distributed algorithm with replanning and the centralized algorithm with replanning are the best choices, and the independent unit MAS algorithm is the worst choice.

6.2 When Using the Concentrated Selection Rule of Agents

This subsection shows the simulation result when using the concentrated selection rule of agents. Figure 6 shows the average number of agent failures and Fig. 7 shows the average number of successful repairs.

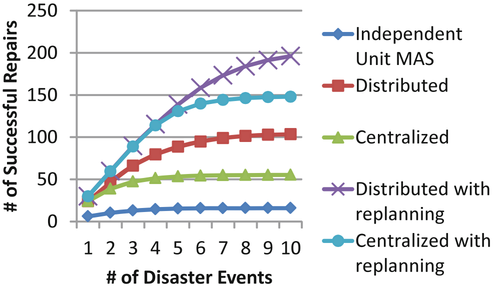


Fig. 7. Average number of successful repairs when using the concentrated selection rule of agents (Balanced deployment of unit MASs)

Now, the differences between the centralized algorithm with/without replanning and the distributed algorithm with/without replanning are clear. The distributed algorithm without replanning becomes better than the centralized algorithm without replanning from the second disaster event. The distributed algorithm with replanning becomes better than the centralized algorithm with replanning from the fifth disaster event. This is because the centralized algorithms work in the same way as the independent unit MAS algorithm after the failure of the top manager agent.

In summary, in this simulation scenario, the distributed algorithm with replanning is the best choice, and the independent unit MAS algorithm is the worst choice.

7 Simulation 2 (Deployment of Inexpensive Unit MASs)

This section shows the simulation results of another pattern of unit MASs where many inexpensive unit MASs are deployed. Because of the budget problem, it is sometimes difficult to install expensive unit MASs that have good sensing/repairing capabilities. A solution to this problem is to use many inexpensive unit MASs. Similar ideas can be found in such areas as crowd sourcing [14], swarm robotics [29], and sensor networks [32].

Table 6 shows the number of each unit MAS that is used for the simulation in this section. The numbers of UMAS 0, . . . , UMAS 6 are 1, 0, 0, 0, 0, 0, 152. Note that from Tables 2 and 6, we can see that performance of UMAS 6 is the lowest in terms of sensing and repairing. Therefore, many inexpensive unit MASs are deployed in this scenario. UMAS 0 is necessary to evaluate the centralized algorithms. The total

Table 6. The number of unit MASs (Deployment of inexpensive unit MASs)

Type of unit MAS	UMAS 0	UMAS 1	UMAS 2	UMAS 3	UMAS 4	UMAS 5	UMAS 6
# of unit MASs	1	0	0	0	0	0	152

number[3] of these unit MASs is 153 (=1 + 0 + 0 + 0 + 0 + 0 + 152). Each UMAS has exactly one manager agent and the total number of manager agents is 153. Each UMAS has exactly one sensing agent except UMAS 0 and the total number of sensing agents is 152. The number of action-execution agents in UMAS 6 is 1 and the total number of action-execution agents is 152. The number of initial resources in UMAS 6 is 3 and the total number of initial resources is 456 (=152 * 3), which is the same as the total number of initial resources of Simulation 1 in the previous section.

We conducted simulations 1000 times using different random seeds. We show the results in terms of the number of agent failures and successful repairs. We show the simulation results in the following two subsections. In Sect. 7.1, we show the simulation results when using the random selection rule of agents. In Sect. 7.2, we show the simulation results when using the concentrated selection rule of agents.

7.1 When Using the Random Selection Rule of Agents

This subsection shows the simulation results when using the random selection rule of agents. Figure 8 shows the average number of agent failures and Fig. 9 shows the average number of successful repairs. Reducing agent failures is the top priority but it depends on successful repairs.

In Figs. 8 and 9, the centralized algorithm and the distributed algorithm are much better than the independent unit MAS algorithm. These two algorithms become even better when combined with a replanning capability because replanning covers unsuccessful repair actions. However, in Figs. 8 and 9 we cannot clearly recognize much difference between the centralized algorithm with/without replanning and the distributed algorithm with/without replanning. These results are almost the same as the results in the previous section. However, it is interesting to see that the number of average agent failures becomes

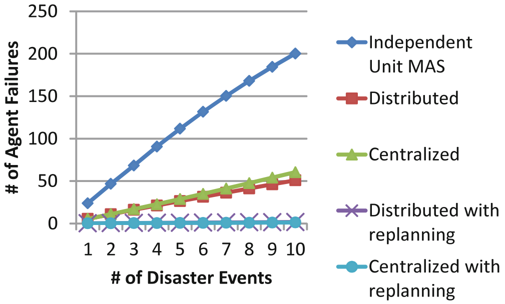


Fig. 8. Average number of agent failures when using the random selection rule of agents (Deployment of inexpensive unit MASs)

---

[3] Compared with [3], the number (153) of unit MASs in Simulation 2 is much larger. In fact, the maximum number of frigates (= unit MASs) in [3] is only 10.

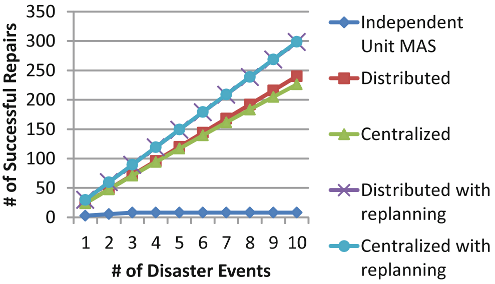


Fig. 9. Average number of successful repairs when using the random selection rule of agents (Deployment of inexpensive unit MASs)

nearly 0 when combined with a replanning capability despite the fact that high performance unit MASs are not used, which is not the case in Fig. 4 in the previous section.

In summary, in this simulation scenario, the distributed algorithm with replanning and the centralized algorithm with replanning are the best choices, and the independent unit MAS algorithm is the worst choice.

7.2 When Using the Concentrated Selection Rule of Agents

This subsection shows the simulation result when using the concentrated selection rule of agents. Figure 10 shows the average number of agent failures and Fig. 11 shows the average number of successful repairs.

Now, the differences between the centralized algorithm without replanning and the distributed algorithm without replanning are clear. The distributed algorithm without replanning becomes better than the centralized algorithm without replanning from the second disaster event. This is because the centralized algorithms work in the same way as the independent unit MAS algorithm after the failure of the top manager agent.

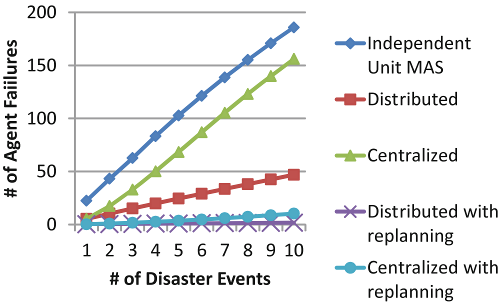


Fig. 10. Average number of agent failures when using the concentrated selection rule of agents (Deployment of inexpensive unit MASs)

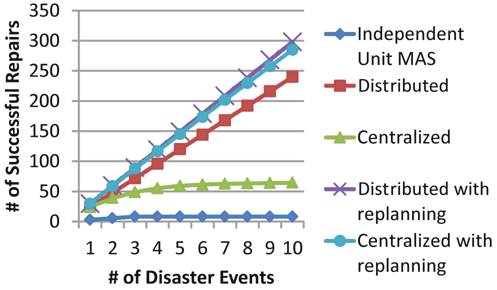


Fig. 11. Average number of successful repairs when using the concentrated selection rule of agents (Deployment of inexpensive unit MASs)

In Figs. 10 and 11, it seems that the distributed algorithm with replanning is slightly better than the centralized algorithm with replanning. However, the differences between the centralized algorithm with replanning and the distributed algorithm with replanning are not so clear as we expected. This is because replanning is very effective and the number of agent failures becomes nearly 0 in both algorithms.

In summary, in this simulation scenario, the distributed algorithm with replanning and the centralized algorithm with replanning are the best choices, and the independent unit MAS algorithm is the worst choice.

8 Discussion

As discussed in the introduction, some repair-task-allocation algorithms are compared in [3], including a no-coordination algorithm (≒ independent unit MAS algorithm), a zone defence coordination algorithm, a contract net algorithm, a simple centralized coordination algorithm, and another central coordination algorithm of [4]. In their test scenarios, the no-coordination algorithm was the best in terms of the number of hits (≒ successful repairs). On the contrary, the independent unit MAS algorithm (≒ no-coordination algorithm) was the worst in our simulation scenarios.

The main reason is that the total number of threats (≒ causes of future agent failures) is small (1 to 10) in the scenario of [3]. In addition, in [3], each ship (≒ unit MAS) tries to remove all the threats using multiple resources in the no-coordination algorithm whereas each threat is assigned to only one ship in the coordination algorithms and replanning is not an option even when it fails to remove the threat. Therefore, action failures lead to agent failures in the coordination algorithms whereas they do not in the no-coordination algorithms.

In the case of our simulation scenario, 10 disaster events repeatedly happen, each of which creates 30 causes of future action failures, and the total number of causes of future agent failures is 300. In this case, without coordination among unit MASs, many causes of future action failures are unattended and not repaired. Furthermore, in the independent unit MAS algorithm, each unit

MAS soon runs out of resources at early stages and cannot continue repairing afterwards. Therefore, in our much severer disaster scenario, the independent unit MAS algorithm is the worst choice and coordination among unit MASs is vital. This result is completely different from the simulation result of [3].

Moreover, we found that replanning reinforces the coordination algorithms in our severe simulation scenarios. We also confirmed that the distributed algorithm is better than the centralized algorithm when the concentrated selection rule is applied.

As we can understand from this discussion, different scenarios can lead to different conclusions. Therefore, with other situations in mind, we intend to conduct more simulations and test more algorithms. This is our future work.

9 Conclusions

In this paper, five algorithms for task allocations were compared by means of multi-agent simulation: the independent unit MAS algorithm, the centralized algorithm, the distributed algorithm, the centralized algorithm with replanning, and the distributed algorithm with replanning. In our severe simulation scenarios, 300 causes of future agent failures are created simultaneously and consecutively. We tested these algorithms for two typical deployment patterns of unit MASs (balanced deployment of unit MASs and deployment of inexpensive unit MASs) and for two different selection rules of agents (random selection rule of agents and concentrated selection rule of agents). Simulation was conducted 1000 times for each case and algorithm. In terms of the average number of agent failures and successful repairs, we arrived at the following conclusions:

- The centralized algorithm and the distributed algorithm are always better than the independent unit MAS algorithm in our simulation scenarios. This means that coordination among unit MASs is very effective, which is completely different from the conclusion reported in [3]. The main reason for this is that many causes of future agent failures are created simultaneously and consecutively in our simulation scenarios.
- When the random selection rule is applied, the distributed algorithm with/without replanning is slightly better than the centralized algorithm with/without replanning in our simulation scenarios. However, the difference is not as much as we expected.
- The centralized algorithm and the distributed algorithm become even better when they are combined with replanning in our simulation scenarios. This means that replanning reinforces the coordination algorithms because repair actions sometimes fail.
- The distributed algorithm becomes better than the centralized algorithm when the concentrated selection rule is applied unless the number of agent failures is nearly kept to 0. This means that the distributed algorithm is effective for unbalanced occurrences of causes of future agent failures. On the other hand, the centralized algorithm is not robust when the top manager agent is vulnerable.

- The distributed algorithm with replanning is the best choice, and the independent unit MAS algorithm is always the worst choice in any case in our simulation scenarios where the number of causes of future agent failures is large. However, as reported in [3], the independent unit MAS algorithm becomes effective when the number of causes of future agent failures is small compared with the number of action-execution agents, and there are enough resources for repairing.

In future work, we are considering the following directions:

- We intend to study the case where manager agents sometimes receive unreliable information of causes of future agent failures and make wrong decisions.
- We intend to improve the algorithms so that we can reduce the number of agent failures. For example, as reported in [13], we have recently improved the algorithm by combining the independent unit MAS algorithm and the distributed algorithm with replanning. We are still trying to improve the algorithms further.
- We intend to conduct simulations in more detail for our target application. For this purpose, it is necessary to combine the MAS controller of our algorithms and the domain-specific simulator of our target application.
- Because different scenarios might lead to different conclusions, we intend to consider other situations and find better algorithms. For example, in one scenario, the network speed between manager agents of different unit MASs might slow or the network might be cut off. In another scenario, the allocation of repair tasks might be corrected by a human operator. Although we use 8 different kinds of unit MASs, we intend to test our algorithms in more heterogeneous domains. We have recently tested some scenarios in [13] where different causes of future agent failures are tested, and failure probabilities and periods of time to failures are different.

References

1. Ahmed, A., Patel, A., Brown, T., Ham, M., Jang, M.W., Agha, G.: Task assignment for a physical agent team via a dynamic forward/reverse auction mechanism. In: Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems, pp. 311–317 (2005)
2. Beaumont, P., Chaib-draa, B.: Multiagent coordination techniques for complex environments: the case of a fleet of combat ships. In: Proceedings of the International Command and Control Research and Technology Symposium (2004)
3. Beaumont, P., Chaib-draa, B.: Multiagent coordination techniques for complex environments: the case of a fleet of combat ships. IEEE Trans. Syst. Man Cybern. Part C **37**(3), 373–385 (2007)
4. Brown, C., Fagan, P., Hepplewhite, A., Irving, B., Lane, D., Squire, E.: Real-time decision support for the anti-air warfare commander. In: Proceedings of International Command and Control Research and Technology Symposium (2001)
5. Brown, C., Lane, D.: Anti-air warfare co-ordination - an algorithmic approach. In: Proceedings of International Command and Control Research and Technology Symposium (2000)

6. Chapman, A., Micillo, R., Kota, R., Jennings, N.: Decentralised dynamic task allocation: a practical game-theoretic approach. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, pp. 915–922 (2009)
7. Chen, J., Yang, J., Ye, G.: Auction algorithm approaches for dynamic weapon target assignment problem. In: Proceedings of International Conference on Computer Science and Network Technology, pp. 402–405 (2015)
8. Choi, H.L., Brunet, L., How, J.: Consensus-based decentralized auctions for robust task allocation. IEEE Trans. Robot. **25**(4), 912–926 (2009)
9. Dawe, M.: Beyond the Kung-Fu circle: a flexible system for managing NPC attacks. In: Game AI Pro: Collected Wisdom of Game AI Professionals, Chap. 28, pp. 369–375 (2013)
10. Gerkey, B., Matarič, M.: Sold!: auction methods for multirobot coordination. IEEE Trans. Robot. Autom. **18**(5), 758–768 (2002)
11. Guessoum, Z., Briot, J., Faci, N., Marin, O.: Toward reliable multi-agent systems: an adaptive replication mechanism. Multiagent Grid Syst. **6**(1), 1–24 (2010)
12. Hayashi, H.: Comparing repair-task-allocation strategies in MAS. In: Proceedings of International Conference on Agents and Artificial Intelligence, vol. 1, pp. 17–27 (2017)
13. Hayashi, H.: Integrating decentralized coordination and reactivity in MAS for repair-task allocations. In: Proceedings of International Conference on Practical Applications of Agents and Multi-Agent Systems, pp. 95–106 (2017)
14. Hirth, M., Hoßfeld, T., Tran-Gia, P.: Anatomy of a CrowdSourcing platform - using the example of microworkers.com. In: Proceedings of International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 322–329 (2011)
15. Johansson, F., Falkman, G.: Real-time allocation of firing units to hostile targets. J. Adv. Inf. Fusion **6**(2), 187–199 (2011)
16. Lagoudakis, M., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A., Jain, S.: Auction-based multi-robot routing. In: Proceedings of International Conference on Robotics: Science and Systems, pp. 343–350 (2005)
17. Macarthur, K., Stranders, R., Ramchurn, S., Jennings, N.: A distributed anytime algorithm for dynamic task allocation in multi-agent systems. In: Proceedings of AAAI Conference on Artificial Intelligence, pp. 701–706 (2011)
18. Mi, Z., Yang, Y., Ma, H., Wang, D.: Connectivity preserving task allocation in mobile robotic sensor network. In: Proceedings of IEEE International Conference on Communications, pp. 136–141 (2014)
19. Nair, R., Ito, T., Tambe, M., Marsella, S.: Task allocation in the rescue simulation domain: a short note. In: RoboCup 2001: Robot Soccer World Cup V, pp. 751–754 (2002)
20. Okimoto, T., Schwind, N., Clement, M., Riberio, T., Inoue, K., Marquis, P.: How to form a task-oriented robust team. In: Proceedings of International Conference on Autonomous Agents and Multiagent Systems, pp. 395–403 (2015)
21. Pujol-Gonzalez, M., Cerquides, J., Farinelli, A., Meseguer, P., Rodriguez-Aguilar, J.: Efficient inter-team task allocation in RoboCup rescue. In: Proceedings of International Conference on Autonomous Agents and Multiagent Systems, pp. 413–421 (2015)
22. Rahimzadeh, F., Khanli, L., Mahan, F.: High reliable and efficient task allocation in networked multi-agent systems. Auton. Agent Multi-Agent Syst. **29**(6), 1023–1040 (2015)

23. Ramchurn, S., Farinelli, A., Macarthur, K., Jennings, N.: Decentralized coordination in RoboCup rescue. Comput. J. **53**(9), 1447–1461 (2010)
24. Ramchurn, S., Wu, F., Jiang, W., Fischer, J., Reece, S., Roberts, S., Rodden, T., Greenhalgh, C., Jennings, N.: Human-agent collaboration for disaster response. Auton. Agent Multi-Agent Syst. **30**(1), 82–111 (2016)
25. dos Santos, F., Bazzan, A.: Towards efficient multiagent task allocation in the RoboCup rescue: a biologically-inspired approach. Auton. Agent Multi-Agent Syst. **22**(3), 465–486 (2011)
26. Smith, R.: The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Trans. Comput. C **29**(12), 1104–1113 (1980)
27. Straatman, R., Verweij, T., Champandard, A., Morcus, R., Kleve, H.: Hierarchical AI for multiplayer Bots in Killzone 3. In: Game AI Pro: Collected Wisdom of Game AI Professionals, Chap. 29, pp. 377–390 (2013)
28. Suárez, S., Quintero, C., de la Rosa, J.: Improving tasks allocation and coordination in a rescue scenario. In: Proceedings of European Control Conference, pp. 1498–1503 (2007)
29. Tan, Y., Zheng, Z.-Y.: Research advance in swarm robotics. Defence Technol. **9**(1), 18–39 (2013)
30. Vallejo, D., Remagnino, P., Monekosso, D., Jiménez, L., González, C.: A multi-agent architecture for multi-robot surveillance. In: Proceedings of IEEE International Symposium on Parallel and Distributed Processing with Applications, pp. 11–18 (2009)
31. Xin, B., Chen, J., Zhang, J., Dou, L., Peng, Z.: Efficient decision making for dynamic weapon-target assignment by virtual permutation and Tabu search heuristics. IEEE Trans. Syst. Man Cybern. Part C **40**(6), 649–662 (2010)
32. Yick, J., Mukherjee, B., Ghosal, D.: Wireless sensor network survey. Comput. Netw. **52**(12), 2292–2330 (2008)
33. Young, B.: Future integrated fire control. In: Proceedings of International Command and Control Research and Technology Symposium (2005)

Author Index

Barile, Francesco 25
Bödelt, Sebastian 193

Cervone, Francesco 25
Çürüklü, Baran 221

Driss, Olfa Belkahla 93

Ekström, Mikael 221

Frasheri, Mirgita 221
Fujita, Wataru 48
Fukui, Ken-ichi 48

Ghédira, Khaled 93

Hayashi, Hisashi 253
Heudin, Jean-Claude 143

Jakubův, Jan 66

Kantert, Jan 193
Komenda, Antonín 66
Kropat, Erik 116

Lopes Cardoso, Henrique 164

Marinheiro, João 164
Meyer-Nieberg, Silja 116
Montazery, Mojtaba 1
Moriyama, Koichi 48
Müller-Schloer, Christian 193

Nouri, Houssem Eddine 93
Numao, Masayuki 48

Rossi, Silvia 25

Sick, Bernhard 193

Tomforde, Sven 193
Tožička, Jan 66

Wilson, Nic 1