Mälardalen University Doctoral Thesis
No.314

Modelling and Control of the Collaborative Behavior of Adaptive Autonomous Agents

Mirgita Frasheri

2020

School of Innovation Design and Engineering
Mälardalen University
Västerås, Sweden

ISSN 1651-4238
ISBN 978-91-7485-468-8
Printed by Eprint AB 2018
Distribution: Mälardalen University Press

Mamit, babit, Titi dajës,
nënë Vitores, dhe gjysh Rizait.

“Be honest, frank and fearless
and get some grasp of the real values of life...
Read some good, heavy,
serious books just for discipline:
Take yourself in hand and master yourself.”

W.E.B Du Bois

Acknowledgements

I dedicate this thesis to my family, the source of inspiration and strength without which I wouldn't be here today. To my mom and dad for being there with me every step of the way. You are, and always will be my reference point; all the important things in life I have learnt from you. To my granny and grandpa for making my childhood joyous, and being the heroes that every kid needs, even more so in adulthood. To my uncle for taking the time over the years to tell me about history, science, arts, and especially good music. Thank you for everything you all have done, and keep doing for me; for giving me courage, and shelter with no reserve.

I would also like to thank friends, colleagues, and teachers, who are, and have been part of my life over the years.

To my lifelong friends, Eda, Nilda, Pami, Laura, for being there for me, cheering for me, cracking me up when morale is down, sharing with me your time, stories, and all those moments of life, big and small. To Ina, for always being in my corner in these 15+ years; I'm not even counting anymore. I have always felt that in you I've found a kindred spirit. Thank you for everything. There are no words good enough to express what our friendship means to me.

To the "grupi akademia" for those hours on end coffee breaks, where we'd discuss about everything, and bicker endlessly; with some of you more than others. In a way, I think, a bit of the merit for getting me out of my shell goes to you.

To the MDH gang. We've had some amazing times together. Be it in our unforgettable trips to Russia, Albania, Northern Spain, and Bosnia, or during our many fikas, dinners, walks, movie nights, game nights, badminton, tennis. For teaching me to ski, play ping pong, and almost skate. For our guitar sessions which have given me immense joy. For all the chocolates and cookies left on my desk, for Joey Ringo and other shark related memorabilia, and a good solid amount of pranks – yes Afshin, I am looking at you. Some of you I really

have to blame, or thank, for tv-series obsessions I have acquired, this last year especially.

To all office-mates I've had over the years, especially Lan Anh – long time sufferer – it has been a pleasure. To the Robotics gang, Carl, Fredrik, Martin and all others, for the support, perfectly timed dry humour, and good fun in our trips and hikes, Friday lunches, or while having one or too many coffees at c2.

To Branko, it has been a pleasure and an honour being your friend, working, and sharing an office with you. You always show up for me, I never once had to ask. I feel that I never fully expressed my gratitude, so here it is, in writing. Thank you. To Aida, for being kind, supportive, and looking after me; also for all the good talks we've had, be it while watching a beach-volley match, or while taking in the sun at a bench at dc. Thank you for being my friend.

To my supervisors. Micke, for your support and positivity, Baran, for surviving those first drafts, and Ale for all the discussions, support and encouragement, for always pushing me to build my own way and to believe a little more in myself.

To my co-authors. Lukas E. thank you for all the questions, and discussions we've had; Eddie, never forget when you thought I was doing my PhD in electronics; José C.G. for working alongside me till the deadline, be it after hours or weekends, Eva G.P. for the feedback, Cristina U. for the continuous and inspiring discussions on my work, and to all you, together with Inma and Joaquin, for having me in Malaga; Václav, especially for making my messy sketches understandable by humans; and all others.

To Prof. Mondi, the best math teacher, or best teacher, period, I have ever had. To the professors at the Polytechnic of Tirana, Elinda K. and Frederik P. among others, for your guidance, support, and positivity. To the teachers and professors at MDH, I have learnt a lot either through the courses or the discussions we've had over the years.

To the administration gang. Carola, Jenny, Malin, and all others, for the laughs, and positivity, and good spirit. For making life so much easier, and simply better. Finally, to Susanne and Radu for your continuous support, from those early Euroweb+ days till now.

Sammanfattning

Forskning om autonoma agenter och fordon har tagit fart under de senaste åren, vilket återspeglas i den omfattande mängden litteratur, och investeringar som gjorts av de stora aktörer i branschen, i utvecklingen av produkter såsom självkörande bilar. Dessutom förutses det att dessa system kommer att kontinuerligt kommunicera och samarbeta med varandra för att anpassa sig till dynamiska omständigheter och oförutsebara händelser, och som ett resultat av detta kommer de att uppfylla sina mål ännu mer effektivt. Underlättandet av ett sådant dynamiskt samarbete och modellering av interaktioner mellan olika aktörer (programvaruagenter, människor) är fortfarande en öppen utmaning.

Denna avhandling tar upp problemet med att möjliggöra för ett dynamiskt samarbete genom att undersöka automatiserad justering av autonomin hos olika agenter, kallad Adaptive Autonomy (AA). En agent är i detta sammanhang en mjukvara som kan bearbeta och reagera på sensordata i den miljö där den är belägen och har dessutom möjlighet att utföra åtgärder autonomt. I detta arbete påverkas agenternas AA av deras villighet att interagera med andra agenter, som fångar agentens egenskaper i att ge och be om hjälp, baserat på olika faktorer som representerar agentens tillstånd och dess intressen. AA-metoden för samarbete används i två olika domäner: (i) att hitta och följa rörliga objekt samt (ii) täckningsproblemet för mobila trådlösa sensornätverk. I båda fallen jämförs den föreslagna metoden med state of the art metoder. Dessutom bidrar avhandlingen på en konceptuell nivå genom att kombinera och integrera AA-strategin - som är rent distribuerad - med en högnivå-uppdragsplanerare för att utnyttja förmågan att hantera lokala och kontingenta problem genom AA-strategin, samtidigt som man minimerar förfrågningarna om en omplanering till uppdragsplaneraren.

Abstract

Research on autonomous agents and vehicles has gained momentum in the past years, which is reflected in the extensive body of literature and the investment of big players of the industry in the development of products such as self-driving cars. Additionally, these systems are envisioned to continuously communicate and cooperate with one another in order to adapt to dynamic circumstances and unforeseeable events, and as a result will they fulfil their goals even more efficiently. The facilitation of such dynamic collaboration and the modelling of interactions between different actors (software agents, humans) remains an open challenge.

This thesis tackles the problem of enabling dynamic collaboration by investigating the automated adjustment of autonomy of different agents, called Adaptive Autonomy (AA). An agent, in this context, is a software able to process and react to sensory inputs in the environment in which it is situated in, and is additionally capable of autonomous actions. In this work, the collaborative adaptive autonomous behaviour of agents is shaped by their willingness to interact with other agents, that captures the disposition of an agent to give and ask for help, based on different factors that represent the agent's state and its interests. The AA approach to collaboration is used in two different domains: (i) the hunting mobile search problem, and (ii) the coverage problem of mobile wireless sensor networks. In both cases, the proposed approach is compared to state-of-art methods. Furthermore, the thesis contributes on a conceptual level by combining and integrating the AA approach – which is purely distributed – with a high-level mission planner, in order to exploit the ability of dealing with local and contingent problems through the AA approach, while minimising the requests for a re-plan to the mission planner.

Abstrakt

Numri i kërkimeve shkencore mbi agjentët dhe automjetet autonome ka pësuar rritje vitet e fundit, fakt i pasqyruar në një literaturë të gjerë shkencore dhe në përfshirjen e lojtarëve të mëdhenj të industrisë në zhvillimin e produkteve të tilla si makinat vetë-drejtuese. Për më tepër, parashikohet që këto sisteme të komunikojnë dhe bashkëpunojnë vazhdimisht me njëri-tjetrin në mënyrë që tu përshtaten rrethanave të ndryshueshme dhe ngjarjeve të paparashikueshme gjatë ekzekutimit të tyre, në mënyrë që të përmbushin qëllimet e tyre me efikasitet. Realizimi i një bashkëpunimi të tillë dinamik dhe modelimi i ndërveprimeve ndërmjet akotrëve të ndryshëm (agjentë softueri, njerëz) mbetet një sfidë e hapur në fushën e sistemeve autonome.

Kjo tezë trajton problemin e realizimit të një bashkëpunimi dinamik mes agjentëve, përmes investigimit së autonomisë adaptive që mundëson rregullimin e automatizuar të autonomisë së agjentëve. Një agjent, në këtë kontekst, është një softuer i aftë të proçesojë dhe të reagojë ndaj inputeve që vijnë nga mjedisi, dhe është gjithashtu i aftë për veprime autonome. Sjellja autonome adaptive e agjentëve karakterizohet nga gatishmëria e tyre për të ndërvepruar me agjentë të tjerë, në formën e kërkesës ose dhënies së ndihmës, si rrjedhojë e ndikimit të faktorëve të ndryshëm që përfaqësojnë gjendjen dhe interesat e agjentëve. Metoda e propozuar për realizimin e autonomisë adaptive është përdorur në: (i) problemin e mbulimit së shënjestrave me κ-agjentë, dhe (ii) problemin e mbulimit në rrjetat me sensorë të lëvizshëm. Në të dy rastet, qasja e propozuar është krahasuar me metoda të njohura në secilën fushë. Për më tepër, teza kontribuon në një nivel konceptual duke kombinuar dhe integruar qasjen me autonominë adaptive – me karakter të shpërndarë – me një planifikues të centralizuar të nivelit të lartë, në mënyrë që të shfrytëzohet autonomia adaptive për zgjidhjen e problemeve në nivel vendor, me qëllim minimizimin e përfshirjes së planifikuesit të centralizuar.

List of Publications

Papers Included in the PhD Thesis[1]

1. Paper A: "TAMER: Task Allocation in Multi-robot Systems Through an Entity-Relationship Model". Branko Miloradović, Mirgita Frasheri, Baran Çürüklü, Mikael Ekström, and Alessandro V. Papadopoulos. 22nd International Conference on Principles and Practice of Multi-Agent Systems (PRIMA'19).

2. Paper B: "Adaptive Autonomy in a Search and Rescue Scenario". Mirgita Frasheri, Baran Çürüklü, Mikael Ekström, and Alessandro V. Papadopoulos. 12th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO'18).

3. Paper C: "GLocal: A Hybrid Approach to the Multi-Agent Mission Re-Planning Problem". Mirgita Frasheri, Branko Miloradović, Baran Çürüklü, Mikael Ekström, and Alessandro V. Papadopoulos. Technical Report[2].

4. Paper D: "Modeling the Willingness to Interact in Cooperative Multi-Robot Systems". Mirgita Frasheri, Lukas Esterle, and Alessandro V. Papadopoulos. 12th International Conference on Agents and Artificial Intelligence (ICAART'20).

5. Paper E: "Adaptive Autonomy in Wireless Sensor Networks". Mirgita Frasheri, José Cano-García, Eva González-Parada, Baran Çürüklü, Mikael

[1] The papers have been reformatted to comply with the doctoral thesis template.

[2] Submitted at the 1st International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS'20).

Ekström, Alessandro V. Papadopoulos, and Cristina Urdiales. 19th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'20).

Additional Peer-Reviewed Publications, not Included in the PhD Thesis

1. "Test Agents: The Next Generation of Test Cases". Eduard Paul Enoiu, Mirgita Frasheri. 2nd IEEE Workshop on NEXt level of Test Automation (NEXTA 2019).

2. "Analysis of Perceived Helpfulness in Adaptive Autonomous Agent Populations". Mirgita Frasheri, Baran Çürüklü, Mikael Ekström. LNCS Transactions on Computational Collective Intelligence (LNCS TCCI 2018). Invited journal.

3. "Comparison Between Static and Dynamic Willingness to Interact in Adaptive Autonomous Agents". Mirgita Frasheri, Baran Çürüklü, Mikael Ekström. 10th International Conference on Agents and Artificial Intelligence (ICAART'18).

4. "Algorithms for the Detection of First Bottom Returns and Objects in the Water Column in Side-Scan Sonar Images". Mohammed Al-Rawi, Fredrik Elmgren, Mirgita Frasheri, Baran Çürüklü, Xin Yuan, José-Fernán Martínez-Ortega, Joaquim Bastos, Jonathan Rodriguez, Marc Pinto. OCEANS'17 conference at the AECC.

5. "An Optimized, Data Distribution Service-Based Solution for Reliable Data Exchange Among Autonomous Underwater Vehicles". Jesús Rodríguez-Molina, Sonia Bilbao, Belén Martínez, Mirgita Frasheri, and Baran Çürüklü. Sensors 17, no. 8 (2017): 1802.

6. "Failure Analysis for Adaptive Autonomous Agents using Petri Nets". Mirgita Frasheri, Lan Anh Trinh, Baran Çürüklü, Mikael Ekström. 11th International Workshop on Multi-Agent Systems and Simulation (MAS&S'17).

7. "Towards Collaborative Adaptive Autonomous Agents". Mirgita Frasheri, Baran Çürüklü, Mikael Ekström. 9th International Conference on Agents and Artificial Intelligence 2017 (ICAART'17).

Contents

I

Thesis

Chapter 1

Introduction

Multi-agent systems (MASs) have been studied extensively in the past decades, and have been used to solve a variety of problems, in both industry and academia, e.g. market simulation, monitoring, and system diagnosis, among others [1, 2]. With respect to logistics, MASs have been used to solve problems such as planning and control, task allocation, negotiation, etc. On the other hand, telecommunication companies, have led the innovation in MAS technologies in their own sector, e.g. early work by *Telecom Italia* laid the foundations for the creation of the JADE (JAVA Agent DEvelopment Framework) agent platform [3], used afterwards for the development of a mediation layer between their support systems and their network equipment and functionalities [4].

Nevertheless, a MAS approach is not a universal solution, and is only suitable in specific scenarios [2]. First and foremost, MASs are appropriate for those problems where different parties are involved, with potentially different goals and interests, that should be aligned. Second, MASs are a useful approach to tackle problems that can be divided into independent or parallel tasks, potentially increasing the computational speed. Third, MASs can be used to increase the robustness to failures of the overall system, and to provide a more graceful degradation to the system. Fourth, a MAS is inherently modular and distributed, and therefore scalable compared to completely monolithic solutions. Additionally, embodied agents, e.g. physical robots, that are able to distribute geographically, create opportunities for use in domains such as search and rescue, given that the robots coordinate properly.

An agent in a MAS is usually defined as a software able to process and

react to sensory inputs in its environment, while additionally being capable of autonomous actions [5]. Interest in the development of autonomous systems has increased in the past years, both from academia and industry [6–10]. As a result, researchers have tackled several issues related to autonomous behaviour including its definitions [11, 12], changing autonomy levels [6, 7], human-like teamwork and collaboration [13–15], placing the human back in the loop and specifying his/her role in the interplay among agents and robots, as well as ethical concerns that have been gaining more and more attention at a rapid pace [16–19]. On the other hand, big industry players such as Google and Tesla, are shaping the state-of-practice with the realisation of self-driving cars. In this context, five autonomy levels have been identified [20], spanning from no automation to full self-driving automation, in which the driver is not expected to keep control of the vehicle.

These autonomous systems are envisioned to collaborate with one another as a result of continuous adaptation to a dynamic environment and unforeseen events, which can have a negative impact on their capability to fulfil given goals and objectives. Modelling interactions between agents such that they are able to collaborate with one another, and dynamically change their autonomy levels, depending on the circumstances, remains an open challenge. Furthermore, these interactions and resulting collaborations impact the autonomy of individual agents, e.g. when agents decide to depend on one another, or are subject to external influences (e.g. receiving help requests), in the context of specific goals and tasks.

This thesis studies adaptive autonomy (AA), a behaviour that allows agents to collaborate by changing their autonomy levels depending on their circumstances in order to achieve their goals. In order to realise AA, a formalism is proposed based on the willingness to interact, which underlies the collaborative behaviour between agents. The willingness to interact covers both facets of interaction between agents, i.e. asking for help and giving help. The utility of the AA approach is investigated in two application domains: (i) mobile wireless sensor networks (MWSN) for extending the longevity of the network, and (ii) hunting mobile search, for monitoring multiple targets, with different viewpoints. The thesis contributes further on a conceptual level by considering how such agent framework could be integrated with centralised approaches, e.g. high-level planning, in order to exploit advantages of both methods such as quick response times, and calculation of optimal solutions, respectively.

The rest of the thesis is structured in two parts. The first part, namely the *Kappa*, contains a comprehensive description of the research activities, goals, and contributions. Whereas, the second part consists in the collection of the

papers included in the thesis.

The chapters in the Kappa are organised as follows. The background is given in Chapter 2, and contains a description of the key concepts relevant in this work such as intelligent agents, MASs, autonomy, and it discusses how this research relates to similar topics. Chapter 3, initially gives the formulation of the problem, the research goal, and the research problems driving the research presented in this dissertation. Afterwards, the five contributions of the thesis are described in detail. The chapter concludes with the research process and method adopted for conducting the research described in the thesis. An overview of the papers included in the thesis is given in Chapter 4. Finally, Chapter 5 lines out the main conclusions, while Chapter 6 identifies interesting directions for future research.

Chapter 2

Background

Agents and autonomy are central to the research described in this thesis. Therefore, in this chapter these concepts are briefly described, starting from the classical definitions of agents given in well-known Artificial Intelligence (AI) literature, building up to intelligent and autonomous agents, and multi-agent systems (MASs). Defining autonomy is not a straightforward matter. In fact there is a plethora of definitions and theories that have been proposed over the past decades, each contributing in a specific way. Here, an attempt has been made to provide a comprehensive view on these different ideas. Finally, the chapter concludes by identifying different research directions with respect to research on autonomous agents, used to provide some context for the contributions presented in this thesis.

2.1 Intelligent Agents

An agent refers to a computer system equipped with sensors and actuators, with which it is able to sense and act in the environment where it is situated, while additionally being able of performing autonomous actions [21, 22]. This definition is easily extended to cover intelligent agents, provided that such autonomous actions are flexible [21], and is considered as the weak notion of agency [23].

The strong notion of agency considers agents from a rather human-like perspective, thus covering aspects such as mental states, beliefs, desires, intentions and so on. Flexibility covers three aspects of agent behaviour: (i) social ability

which refers to agents interacting with one another, (ii) reactivity to the stimuli coming from the environment, and (iii) pro-activity in terms of deciding own goals or courses of action. Wooldridge [24] further argues that a balance between pro-activity (goal-directedness) and reactivity is required by intelligent agents, i.e. neither blindly executing procedures nor continuously reacting to the environment – thus achieving no goals – are desired behaviours. Furthermore, social ability is more than the exchange of messages. In fact, it refers to interactions of entities that are autonomous and have their own goals, which do not necessarily have to overlap. Thus, these agents[1], will have to negotiate and cooperate in order to achieve their goals. Research on agents generally falls in one of three lines of inquiry, identified by Wooldridge and Jennings [25], such as agent theories, agent architectures, and agent languages. Agent theories are concerned with how to conceptualise agents, and can make use of formal models for describing agents and their properties. Agent architectures build upon these agent theories and aim to build concrete agents with software and/or hardware. Agent languages reside at a technical level and target the software tools and languages which can be used to implement agents.

There are three well-known paradigms that target the design of agent architectures: (i) symbolic or deliberative, (ii) reactive, and (iii) hybrid which is simply a combination of both approaches [23]. The symbolic paradigm is connected to classical AI approaches [25], and relies on the physical-symbol hypothesis stated by Newell and Simon: "A physical symbol system has the necessary and sufficient means for general intelligent action" [26]. The physical symbol system is composed of a set of symbols, as well as processes and operations that could be applied to structures of these symbols[2]. Processes and operations themselves can create, modify, reproduce and destroy the symbolic structures. Agents built upon the symbolic paradigm reason on the symbolic representations of their world in order to decide about what beliefs to have and actions to take (e.g. theorem proving). Such reasoning revolves around the beliefs that an agent has about the current state of affairs in its environment and its desired state in the future. On the other hand, in order to achieve such

[1] Agents are sometimes confused with concept of objects in object-oriented methodologies [24]. However, there are several differences at a conceptual level. An agent is assumed to have some autonomy that enables it to decide not to execute a method requested by another agent. Furthermore, the other properties that relate to flexibility as discussed above are not at the core of the object concept. Lastly, an agent generally corresponds to one thread of control.

[2] In most interpretations, the full implications of Newell and Simon's ideas are not captured [27]. Besides running over patterns of symbols, processes can generate processes, as well as patterns can generate patterns, and as a result a physical symbol system, in principle, is able to develop.

desired state, an agent has to engage in practical reasoning (means-ends reasoning or planning). In such architectures, the main problems to be solved are: (i) the representation/reasoning problem, which refers to how knowledge is represented as well as how agents reason with such knowledge, and if they do so in time, and (ii) the transduction problem, that relates to the generation of adequate representations of the world [24]. Issues include the computational complexity of, e.g. theorem proving which affects whether an agent will be able to reason in time, as well as how to create adequate symbolical representations of the real world. Furthermore, for such architectures it is assumed that the environment does not change in a crucial way while an agent is reasoning.

In the mid-eighties, Rodney Brooks, a rather vocal critic of symbolic approaches, argued that intelligence is a result of the interaction of the individual with the environment [28]. Moreover, the environment is its own model, and thus an agent does not need a corresponding symbolical representation [29]. His ideas paved the way for the class of reactive architectures. He proposed the Subsumption architecture [30], where an agent is composed of several behavioural layers, placed one above the other, which compete for execution. Low layers deal with surviving behaviours such as collision avoidance, whereas higher levels can be goal-fulfilling behaviours, and additionally they can be inhibited by lower layers. As it turned out, such agents are simple to build, non computationally expensive and robust, and able of performing some tasks [24]. However, several issues arise that relate to whether (i) the available local information is enough in order to come to a good decision, (ii) they can learn and improve over time, or (iii) more complex behaviours could be achieved. Furthermore, these agents rely on the emergence of intelligence, and the engineering of emergent behaviour is not straightforward. In order to take advantage of the strong points of each paradigm, researchers have been working on hybrid architectures that combine symbolic reasoning and reactivity. The goal is to make use of the sophisticated reasoning that is inherent for symbolic agents, while still having agents that are able to react to a dynamic and changing environment. One example is the TouringMachines architecture [31], composed of layers in charge of planning and reactive behaviour among others. However, such architectures are not conceptually and semantically clear as symbolic architectures. Furthermore, it is not straightforward how these layers should interact with one another [24].

Research on intelligent agents is quite relevant for the study of cognition. Although the work in this thesis does not deal with cognition, it is worthy of note how the paradigms of cognition relate to the classes of intelligent agents. In fact, there are two broad classes of cognitive theories, the symbolic and

emergent [32]. In the former, the basic assumption is that cognition stems from the manipulation of symbols. Example architectures are Soar [33], ACT-R [34] among others. In the latter, cognition stems from systems ability to self-organise, while maintaining their autonomy in the process, e.g. AAR [32], SASE [35], and DARWIN [36] among others. Researchers have pursued the idea of hybrid cognition theories, the aim of which is to bring together the strongest points of each paradigm, e.g. Cerebus [37], and Cog [38] among others.

2.2 Multi-Agent Systems

A group of agents that interact with one another in order to fulfil common tasks, while not necessarily having aligned objectives, is called a multi-agent system (MAS) [39]. MAS are related with distributed problem solving (DPS) research, both sub-fields of distributed artificial intelligence (DAI) [40]. Such relation has been viewed from three perspectives [41]. The first view considers DPS as a subset of MAS, i.e. a MAS system is a DPS if it is possible to assume that agents are benevolent, have common goals, as well as a centralised designer. Furthermore, in a DPS, the cooperation and coordination of nodes is determined during the design, and thus cannot change during run-time [24]. In the second view, MAS research, which focuses on how to build agents with certain properties and is interested in the emergent properties when these agents interact, underlies DPS research. The latter takes agents with desired properties as a given, and focuses on the achievement of external properties by the system as a whole. In the third view, MAS and DPS are considered simply as two different research agendas, in each of which researchers ask sets of different questions. In MAS, the questions relate to how agent are built and how they interact, whereas in the DPS case, the focus is on the external behaviour of the system in terms of performance.

Generally, a MAS is characterised by five main properties [39]: (i) autonomy, i.e. the MAS cannot be controlled by an external party, (ii) knowledge, in terms of skills and beliefs, is distributed, and control is decentralised, and as a result agents need to interact in order to accomplish tasks that are beyond the abilities of a single-individual [42], (iii) agents operate in a parallel and asynchronous way [43], (iv) openness, MAS can be either open or closed systems in terms of the possibility of new individuals coming in during operation, and (v) heterogeneity, where agents in the MAS can be either homogeneous or heterogeneous with respect to one another. Three classes of MAS have been

identified [39]: (i) MAS where agents are in the role of assistants to users and interact with one another on the behalf of these users, (ii) MAS for simulation used to model and study natural phenomena, and (iii) MAS for collective problem-solving.

According to Ferber [44] MAS interaction types can be characterised with respect to their goals, resources, competences, and situation type (Table 2.1). Agents with compatible goals are indifferent to one another as long as they have sufficient resources and competences. As soon as these agents have insufficient resources, competences, or both, there is simple cooperation between them, i.e. coordination between entities that have compatible goals [40]. On the other hand, for agents with incompatible goals, interactions are rather antagonistic. Thus, cooperation requires negotiation between agents, or in the words of Weiss, coordination between entities that do not have aligned goals [40]. In this work, agents are assumed to have compatible goals, while either resources, competencies, or both are insufficient.

2.3 Autonomy

The following paragraphs provide a discussion on the different definitions and models of autonomy, as well as a research map of the different directions of autonomy-centred research.

2.3.1 Definitions and Models

Autonomy has been defined in different ways over the past decades. It is possible to speak of autonomy in terms of two dimensions: (i) self-sufficiency, i.e. the ability of performing tasks without external help, also referred to as independence [11], and (ii) self-directness, i.e. the ability of selecting goals on one's own, respectively referred to as the descriptive and prescriptive dimensions by Bradshaw *et al*. [45]. Vernon uses similar dimensions to characterise robotic autonomy [27], in terms of strength of autonomy (self-sufficiency), i.e. how much a robot can take care of itself for different degrees of uncertainties, and degree of autonomy (goal-directedness), i.e. how much help is needed from the human. On the other hand, autonomy is a relational concept [11], and as such can either be defined with respect to other entities such as oneself, e.g. between agents, also referred to as social autonomy, or between an entity and its environment, i.e. how autonomous one is from the stimuli coming from such environment. Furthermore, social autonomy can be expressed in indepen-

Table 2.1: Ferber's interaction types [44]

	Goals	Resources	Competences	Situation Type	Interaction Category
1	Compatible	Sufficient	Sufficient	Independence	Indifference
2			Insufficient	Simple collaboration	
3		Insufficient	Sufficient	Cluttering	Simple cooperation
4			Insufficient	Coordinated collaboration	
5	Incompatible	Sufficient	Sufficient	Pure individual competition	
6			Insufficient	Pure collective competition	Antagonism (requires negotiated cooperation)
7		Insufficient	Sufficient	Individual conflicts for resources	
8			Insufficient	Collective conflicts for resources	

dence (self-sufficiency) and in agency. The latter considers the autonomy of an agent working on the account of another agent.

Depending on the view, an agent could be self-sufficient to some degree, or have a particular degree to which it is working for other agents, ranging from being exploited, having to obey, helping as a peer and so on [11]. Therefore, it is possible to think in terms of levels of autonomy. In this regard, the ten levels of automation were proposed [46, 47], with the intent of providing a guideline for understanding and building systems that showcase different autonomy levels (Table 2.2). In the lowest level the machine is merely a slave, and will do the bidding of the human operator to the point. Going up in the levels, the machine gets more and more initiative to perform specific tasks autonomously, e.g. in layer 3 the machine can narrow down the selection of alternatives, or in layer 8 where it will only inform the human if explicitly asked to do so. The highest level corresponds to full autonomy and independence from the human.

Many more theories on autonomy have been brought forward, as well as definitions as noted by Vernon [27], such as, adaptive autonomy, adjustable autonomy, basic autonomy, belief autonomy, robotic autonomy, shared autonomy, sliding autonomy, to name a few. In some of these types, while there are several levels of autonomy a system can operate at, the differences lie on the actor that is able to change level, e.g. adjustable autonomy [6], where the human decides the level of autonomy, sliding autonomy [48], where the system can switch between tele-operation and full-autonomy on a task-level, adaptive autonomy[3] [6], where the system itself decides on its own level of autonomy, mixed-initiative interaction [12], where both human and agent are able to change the autonomy level, collaborative control [50], where the human and the agent resolve conflicts through dialogue.

Castelfranchi defines agent autonomy using dependence theory [11]. Dependencies between agents define the autonomy levels they have with respect to one another. These dependencies can be of different granularity, and is dependent on the agent architecture. Assume an agent a_i that intends to complete a task. However, a_i lack some of the means needed for performing such task, e.g. information, know-how, planning abilities, skills, or material resources among others. Further, assume the presence of an agent a_j on which a_i can rely for what is needed for completion of the task. From this perspective, a_i is non autonomous from a_j in that moment in time, with respect to the specific task and the means that a_j is providing. Johnson *et al.* similarly discuss about potential inter-dependencies between different agents, which should be used in

[3] Note that, Vernon [27] maintains another view, discussed by Ziemke [49], in which adaptive autonomy relates to the interaction between complex organisms and the environment.

Table 2.2: The 10 levels of autonomy proposed by Parasuraman *et al.* [47]

Autonomy	Lvl.	Description
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

the design of joint-activity [7, 12].

In this thesis, adaptive autonomous agents are able to decide on their own autonomy levels, as discussed by Hardin and Goodrich [51]. Furthermore, based on Castelfranchi's definition of agent autonomy [11], adaptive autonomy is further defined in terms of changing dependencies between agents. Indeed, at a time τ_1 an agent might be able to complete a task on its own, however at a time τ_2, due to a possible change of circumstances, e.g. sudden drop of the battery level, the agent cannot perform its task on its own. Thus, it will ask for help, and depend on another agent, with respect to which it will not be autonomous, in the context of the particular task.

2.3.2 Related Research Directions

Six main directions of research can be identified in the domain of autonomous agents, as follows: (i) design methodologies, (ii) regulatory systems, (iii) agent architectures, (iv) comparative studies, (v) human-machine interfaces, and (vi) algorithms for changing autonomy levels.

Design Methodologies. Design methodologies target the development of systems that change their autonomy levels. Johnson *et al*. emphasise the importance of considering interdependence between agents early on in the development of systems that should support joint-activity [12]. Agents are considered interdependent if they rely on others during the execution of their tasks. Furthermore, interdependence could be either soft, i.e. relying on others is not necessary but could improve performance, or hard, i.e. relying on others is necessary to complete a task. Subsequently, the same authors proposed a design method based on the interdependence concept, with the aim of providing engineers tools to be used during the design and implementation of systems with adjustable autonomy [7]. The method consists of three steps. Initially, potential interdependence relations are identified. Afterwards, mechanisms are designed that can support them. Lastly, it is investigated how the mechanisms affect the existing interdependent relationships.

Regulatory Systems. Regulatory systems refer to approaches based on policies and societal norms intended to shape agent behaviour. The main role of such systems is to support predictability and bring about coordination between agents [52]. Furthermore, regulatory systems can be used to enable changes of autonomy in a community of agents. An example is the Kaa system [53], which extended an existing policy system (KAoS), and allowed a central coordinator to override or modify policies concerning specific agents during run-time, as a response to the circumstances. The human was involved when Kaa would not be able to make a decision.

Agent Architectures. Agent architectures have been developed in order to provide support teamwork and autonomy level adjustment such as, STEAM [54], DEFACTO [55], and THOR [56]. The STEAM architecture [54] extended Soar [57] by adding support for teamwork through the addition of team-operators [4]. Team operators are used to allow agents to reason about team plans, additionally to their individual plans which do not require teamwork. The proposal also consists of a synchronisation protocol between agents when executing team plans. The DEFACTO framework [55] targets the provision of support for transfers of control in continuous time, while simultaneously resolving human-agent inconsistencies, and making actions interruptible for real-time systems. Team THOR's entry in the DARPA Robotics Challenge [56] consists of a motion framework for a humanoid robot which allows for low-

[4] This approach was based on joint-intention theory [58].

level control, as well as scripted autonomy, i.e. invocation of robot movements by calling predefined scripts, and supporting high-level commands from the user.

Comparative Studies. Other researchers have focused on the investigation of different schemes for switching between autonomy levels. Such studies include comparisons of: (i) static autonomy versus dynamic autonomy [59], and (ii) different implementations of dynamic autonomy, such as AA where agents change their own autonomy, mixed-initiative interaction, and adjustable autonomy [6]. In (i), several decision frameworks are identified such as master-slave, peer to peer and locally autonomous, and switched in between in a series of experiments in order to demonstrate the benefits of dynamic autonomy as compared to static autonomy. Mapping between a decision framework and conditions in the environment was determined beforehand. Indeed, the study's purpose was to motivate the design of systems that are able to change their autonomy. Authors have shown that agents perform better when compared to the cases where there is a single decision-making framework. In (ii), the three different schemes for changing autonomy levels were evaluated through a series of experiments. These schemes were defined as follows. Adjustable autonomy allows the operator to select the appropriate autonomy level, whereas adaptive autonomy enables agents to decide themselves while always trying to maintain the highest level. Mixed-initiative interaction allows both operator and robot to make changes on autonomy levels. Authors consider the cooperation of a human operator with simulated robotic searchers, in the range of 200 such searchers. Specifically, they look into the relation between operator workload, prior knowledge, autonomy scheme, and the impact on performance. They measure the performance through both primary metrics, such as number of individuals found, and secondary metrics, such as cues found in the environment. They show that while adaptive autonomy performs better on the secondary metrics, mixed-initiative interaction reports the highest score for the primary metrics. This comes as a result of the fact that operators, when in possession of relevant information, are able to have more influence on the robots. This is not the case for robots with adaptive autonomy, which cannot take direct commands, and attempt at all times to maintain the highest level of autonomy.

Human-Machine Interfaces. A prominent area of research is human-machine interaction, specifically mechanisms that allow both parties to learn from one another, and change the autonomy as necessary. One such proposal involves enabling a robot to perceive the human's controlling skills, and adjust

its level of autonomy accordingly [60]. Controlling skills involve a variety of aspects such as navigation, manipulation (gripping faculties), and multi-robot coordination. Others allow the user to control a robot at predefined autonomy levels such as: low-level body control, setting navigation way-points or object selection, setting high-level goals such as "bring the can with a soft drink" [61]. Issues concerning human aided coordination of teams of robots which might need assistance occasionally, have also been tackled [62]. Such interfaces need to provide all parties involved an overall view of state of affairs at all times [63, 64] and their design also depends on how much autonomy a system is supposed to have [65].

Algorithms for Changing Autonomy Levels. Several mechanisms for changing autonomy levels have been proposed. One approach consists in a meta-reasoning model based on heuristics that defines how external factors influence the behaviour of an agent [66]. Authors evaluate their approach in a fire-fighter scenario, composed of fire-fighter agents, and a centralised coordinator – effectively a superior to the agents. Every agent has two goals that involve putting out fires, and maintaining health by allocating time to rest. The heuristics that influence the reasoning are:

1. Task urgency. An agent will filter out information not related to a task that has become urgent. Furthermore, a tired agent will increase the urgency for taking a rest.

2. Dedication level. It is assumed that an agent is dedicated to its organisation, and is thus more dedicated to the goals of the organisations than self-set goals. Nevertheless, agents will decrease their dedication if wrong information keeps coming from the coordinator – that represents the organisation – and define their own goals, e.g. which fires to attend to.

It is shown that adaptive behaviour can be achieved through this model, and that it performs better than the cases where no such meta-reasoning is used. Brookshire *et al*. consider changing autonomy at the task level [48] to achieve sliding autonomy. Consequently, an agent might be able to execute some task autonomously, whilst it should be tele-operated for achieving another task. Others, perform a task categorisation before hand, i.e. tasks that could be performed autonomously, and tasks that need supervision from an operator [67]. Thereafter, such selection guides the design of algorithms. Unexpected events

may lead to a human operator taking control and interrupting the on-going activities of a team of robots. In order to provide support for such interruption, an approach based on Coloured Petri Nets has been used to formalise team plans and interruption mechanisms [68].

The research presented in this thesis deals with mechanisms that enable agents to display adaptive autonomous behaviours as a response to their circumstances at any point in time, both from a purely distributed perspective and in combination with centralised components. Furthermore, benefits of adaptive autonomy with respect to static autonomy have also been investigated.

Chapter 3

Research Overview

This section provides an overview of the conducted research, and is structured as follows. Initially, the addressed problem is described, from which the overall research goal of thesis and the investigated research problems are derived. Thereafter, a compact description of the contributions of this thesis is given, as well as the mapping between research problems and contributions. The chapter concludes with an account of the research process, focusing on the adopted evaluation method.

3.1 Problem Formulation

The development of autonomous systems, i.e. systems that are able to make decisions and fulfil tasks on their own, has engaged researchers continuously over the past decades. These systems are envisioned to be fully integrated in everyday life, where they are able to collaborate with one another, as well as with humans – either trained operators, users or simple bystanders – in order to complete some task. Furthermore, such systems might need to collaborate in such a way that they are able to adapt to changing circumstances that can be a result of dynamic environments, changes in the capabilities or conditions for performing the desired tasks. Thus, it is of interest to equip these systems with local mechanisms that allow them to adapt their own autonomy, and collaborate in any context[1].

[1] Several researchers have taken their inspiration from human-like teamwork [13–15].

This thesis tackles the design and evaluation of local mechanisms that enable a system, i.e. an intelligent agent as defined by Wooldridge [21], to collaborate with others as a response to changing circumstances so as to fulfil its goals and optimise its performance, adapting its own autonomy in the process. Thus, the research goal of this thesis is formulated as follows:

Research Goal. *Model and control the adaptive autonomous behaviour of agents in a MAS in order to optimise their performance, as compared to static and reactive strategies, in the context of dynamic environments and circumstances.*

In order to achieve this goal, four research problems **RPs** have been identified, each residing at a specific abstraction level, among survey, conceptual, and problem levels.

RP 1. *Identify the properties characterising MASs and multi-robot task allocation problems (MRTA).*

The first research problem involves understanding and analysing existing research that has tackled MAS systems and MRTA problems, as well as identifying the properties that characterise them. **RP 1** lies at the survey level, and represents the groundwork for the thesis.

RP 2. *Model the adaptive autonomous behaviour of an agent such that it enables agents to change their autonomy levels and collaborate with one another as a response to changing circumstances.*

The second research problem lies at the conceptual level, and is considered from a purely distributed perspective, i.e. the focus is on how agents in a MAS can reason about their disposition to collaborate by asking and giving help to one another. Such reasoning depends on the local state of each agent, e.g. battery levels or skill set among others, and the local information that they have access to at a given point in time. There are neither central components, nor hierarchies assumed between agents.

RP 3. *Integrate adaptive autonomous agents with centralised high-level planning in order to make use of the advantages of both paradigms, i.e. distributed and centralised planning.*

The third research problem lies as well at the conceptual level, and involves investigating the adaptive autonomous behaviour of agents in a wider context, by including in the loop a centralised component such as a high-level planner.

The interest lies in how to integrate these two different approaches, in order to exploit the advantages of both, i.e. optimality of the solution and robustness displayed by the system, respectively.

RP 4. *Apply an adaptive autonomous agent approach to specific problems in order to optimise the performance of a MAS, and investigate the benefits of the proposed approach.*

The fourth research problem lies at the problem level, and focuses on the applicability and efficiency of the proposed approach in solving specific problems within the MAS domain. Furthermore, by selecting well-known problems, it is possible to make comparisons with state-of-art methods and understand the benefits and limitations of the proposed approach.

3.2 Research Contributions

This thesis consists of five contributions (C1-C5) which tackle the research problems defined in Section 9.3. The first contribution is a survey on the taxonomies that target MAS properties and multi-robot task allocation (MRTA) problems, and attempts to provide a common view on the different aspects that characterise MRTA. Contributions C2 and C3 lie at a conceptual level. The second contribution is a model for adaptive autonomy that underlies the collaborative behaviour of agents. Whereas, the third contribution aims to integrate collaborative agents with high-level planning in order to make use of the advantages of both paradigms. Contributions C4 and C5 are problem-oriented, i.e. the focus is on how the proposed model (C2) can be adopted in solving specific problems. The fourth contribution relates to applying the model for agent collaboration in solving the mobile multi-object κ-coverage problem. Finally, the fifth contribution applies such model for extending network longevity and coverage in mobile wireless sensor networks.

C1: Providing a unified view of taxonomies on multi-robot task allocation problems.

Multi-agent and multi-robot systems (MRS) have been subjects of extensive research in the past decades. Their distributed nature allows the representation of different stakeholders, i.e. different interests within the same application, as well as to provide opportunities for boosting performance, scalability, and robustness [2]. In the early 1990s and early 2000s, researchers mainly focused on the properties of MAS and on the collaborative behaviours such as communication, topology, robot group composition, among others [2, 69, 70],

while task allocation played a minor role. The first to shift the focus towards the task allocation problem in MRS was Gerkey and Matarić [71], who proposed three orthogonal dimensions for their classification: (i) single versus multi-task robots, which relates to the ability of a robot to carry out several tasks in parallel, (ii) single versus multi-robot task, which refers to the number of robots needed to perform a task, and (iii) instantaneous versus time-extended assignment, the former allowing only immediate allocations of a new task without consideration of other tasks, and the latter corresponding to allocations which take into account how tasks might arrive in the future, or the whole set of tasks. This taxonomy however, as pointed out by the authors themselves, did not account for dependencies between tasks. Three independent works have built upon the taxonomy proposed by Gerkey and Matarić [71], two of which address the issue of dependencies [72, 73], while the third specifies further the time-extended assignment dimension [74]. Concerning the former, Landen *et al*. proposed to extend the initial taxonomy with the dimensions of unrelated versus related utilities between tasks, and independent versus constrained tasks [72]. Whereas, Korsah *et al*. proposed the degree of inter-relatedness to cover for both of those dimensions, which was based on their classification of tasks [73]. Nunes *et al*. tackled the time-extended assignment and specified it further with time windows, distinguishing between temporal and ordering constraints, and synchronous precedence assignment [74].

Apart from the focus shift between the different works, these taxonomies overlap with respect to the aspects they consider for MAS and MRS. In order to address these overlaps and provide a structured overview of MRTA problems, the TAMER model was proposed (Task Allocation in Multi-Robot Systems through an Entity Relationship Model) in Paper *A*. As the name suggests, the relevant aspects for MRTA problems are modelled through entities, and the relationships between them, according to the Entity-Relationship model. Four such entities have been identified: *Robot*, *Task*, *Environment*, *Mission*, and nine relationships to be described briefly. The robot entity is specified by the capabilities (hardware, software, concurrency), behaviour, state, and id fields. The way in which robots create teams and communicate with one another is captured by the *teamed* and *communicate with* relationships. The environment entity is characterised by properties such as observability, uncertainty, determinism, discreetness, state and id. The interaction between the robot and environment entities is expressed through the *act* relationship. The task entity is defined by fields such as type, required capabilities, interruption, and id. Task decomposition and dependencies are captured by the *decomposed* and *depends on* relationships. The mission entity is characterised by the objec-

tives, constraints, and id. This entity is related to the environment, robot, and task through *deployed*, *includes*, *includes* relationships respectively. The main relationship in TAMER is the *allocation* which binds together a mission with tasks and robots. The benefit of such approach is that it allows to add/remove fields in a principled way, e.g. a new field can be added only if that information cannot be deduced in the current form. Furthermore, with TAMER, it is possible to define multiple missions, allocations, environments, concerning the same agents/robots.

C2: Modelling the adaptive autonomous behaviour of agents through the willingness to interact.
The collaboration between agents in a MAS, typically to complete a task that is beyond the capabilities of a single agent, has an implication on the autonomy of the agents involved. When agents collaborate, they are depending or allowing others to depend on them for achieving a particular task τ. Moreover, such dependencies can change in time. At a time t_1 an agent a_i is able to complete τ on its own, whereas at a time t_2 it needs to rely on another agent a_j due to, e.g. low battery levels. In the latter case, a_i is not autonomous from a_j concerning τ. Therefore, depending on the circumstances, a_i might need to adapt its own autonomy level, and collaborate with others in order to achieve τ. From the perspective of the agent that provides such help, two arguments are made. As long as an agent is able to make its decision on whether to accept or reject a request for help, it is considered autonomous. However, allowing an external entity to influence one's internal state or course of action – a_j in order to help a_i needs to modify its current plans – is limiting the autonomy of an agent. Such dynamics, of depending and assisting others, and in turn adapting one's own autonomy, are relevant for agents that collaborate. In fact, a fully autonomous agent, that neither depends on others nor gets influenced by others is not interesting in the context of a MAS, as it is in practice isolated from the rest [75].

In order to model adaptive autonomous behaviour, which underlies collaborative behaviour, the willingness to interact abstraction was proposed (Paper *B*). This formalism is composed of the willingness to give and ask for help, thus capturing both aspects of an interaction. A mathematical framework for the estimation of the willingness is presented which accounts for factors determined as relevant to whether an agent should ask or give help, considering as well the weight of each factor. In Paper *B*, nine such factors are described that relate to different aspects of an agent such as: battery level, confidence on own knowledge, efficiency of the skill set, accuracy of equipment, quality of

resources, overall performance level, task progress/task trade-off[2], likelihood of succeeding in an environment, and the likelihood of a successful collaboration with another agent. This list of factors is neither meant to be exhaustive, nor necessary for every application domain, as shall be seen in Papers *D* and *E*. Its purpose is to create an understanding of the different factors that can be relevant for the willingness to interact.

The willingness to interact was evaluated in Paper *B* against the following hypothesis:

Hypothesis. *Agents with adaptive autonomy perform better than agents that do not display such behaviour, in the context of a simulated search and rescue scenario.*

To this end, a simulation environment was setup, built upon the Robot Operating System (ROS) middleware. Emergency sites, consisting of fires to be extinguished and people to be evacuated, were generated randomly in 2D space *Z* of given width and height. Three types of agents were implemented: (i) fire-fighters whose goal is to go to an emergency site and put out fires, (ii) ambulances that can take people from the emergency sites after the fires have been put out, and (iii) police officers that scout the area and communicate a detected emergency site. Independently of the role, an agent was designed as a state machine with five states such as *idle*, *execute*, *interact*, *regenerate*, and *out of order*. In *idle* agents perform random walks within *Z* in order to detect an emergency site. Agents shift to *execute* once they have a task to perform (extinguishing a fire, removing people from the site). The willingness to ask for help will determine whether an agent will send a help a request, e.g. in case its resources are out. When a request is received, an agent switches to *interact*, where based on its willingness to give help will send a response to the agent in need. The *regenerate* and *out of order* are auxiliary states. When an agent's battery is below a given threshold it will switch to *out of order*. Whereas when an agent attempts to restore itself, e.g. by recharging, it switches to regenerate. Simulations are run in two modes, static and dynamic, and at two difficulty levels each. In the former, the components of the willingness are set at the beginning of the simulation, and remain unchanged throughout. In the latter, they adapt at every iteration. The difficulty is determined by the intensity of the fires, and number of people trapped, while the initial resources of agents

[2]Note that the progress on a task influences an agent's willingness to ask for help, whereas task trade-off impacts an agent's willingness to give help. The rest of the factors remain the same for both components of the willingness to interact.

remain the same. It was shown that, agents that are able to adapt their willingness, perform better than their static counterparts in terms of: (i) dependent tasks completed, i.e. tasks that were achieved with help from others, and (ii) higher percentage of completed tasks with respect to returns to the base, where an agent refills its resources.

C3: Integration of the agent approach based on the willingness to interact with high-level planning.
Automated planning deals with the allocation of a set of tasks to a group of agents or robots, such that the objectives of the system as a whole are achieved. As unforeseen events occur while agents are executing their respective tasks, initially devised plans are no longer feasible. Consequently, a re-planning process is necessary, i.e. planning with new initial conditions, that takes into consideration the current state of affairs, and reallocates the remaining tasks accordingly. Solutions proposed to tackle the re-planning problem generally fall in either of the following categories: centralised global re-planning, or in distributed local re-planning. Centralised global approaches can produce optimal plans if such exist, as well as account for the overall state of the system, at the cost of being computationally heavy, with the centralised planner representing a single point of failure. Furthermore, while the time under disposal for creating an initial plan is in principle unbound, the same does not hold for re-planning. Distributed approaches on the other hand rely on agent cooperation for solving issues that arise during operation. While there are no guaranties of optimality, these approaches can produce good enough solutions within a shorter time.

In order to exploit the strengths of both approaches, in terms of optimality of solutions and robustness, a hybrid approach was proposed, named GLocal. GLocal was adopted to solve a relaxed version of the Extended Coloured Travelling Salesperson Problem (ECTSP) [76], that does not contain precedence constraints as the original ECTSP. Consider a set of n tasks, $v \in \mathcal{V} := \{v_1, v_2, \ldots, v_n\}$, m agents, $s \in \mathcal{S} := \{s_1, s_2, \ldots, s_m\}$, and k capabilities, $c \in \mathcal{C} := \{c_1, c_2, \ldots, c_k\}$ where $m, n, k \in \mathbb{N}$. Every agent $s \in \mathcal{S}$ is assigned a set of capabilities $\mathcal{C}_s \subseteq \mathcal{C}$. Each task $v \in \mathcal{V}$ requires one capability in order to be successfully completed. A capability matrix of an agent s, $\mathcal{A}_s \in \{0,1\}^{n \times n}$, can be defined as:

$$a_{ijs} = \begin{cases} 1, & f_c(v_i) \in \mathcal{C}_s \wedge f_c(v_j) \in \mathcal{C}_s \\ 0, & \text{otherwise,} \end{cases} \tag{3.1}$$

Agents are allowed to move in a 2D space Z, and are able to communicate with each other with broadcast, without limitation in range. The problem is

formulated as: allocate n tasks to m agents such that the given constraints in terms of agent capabilities and task requirements are respected, in order to minimise the make-span of a mission. The make-span is defined as the duration between the starting time of the first task and end time of the last task over all agents in a mission. GLocal enables agents to attempt a plan reparation through agent negotiation, in the eventuality of an agent failure. In case the process of re-allocating all tasks belonging to the failed agent does not succeed, then the planner is invoked, which re-plans for all tasks and non failed agents based on updated information, e.g. current agent locations.

GLocal was evaluated through computer simulations and compared with a planner-only approach, considering different numbers of failures $n_f \in \{0, 1, 2, 3\}$, and sizes of problem instances given by the number of tasks in a mission $s_{PI} \in \{50, 100, 150\}$. Failures refer to agents that break during execution, and are as a result not able to perform any tasks or communicate with other agents. No failure can make a mission infeasible, i.e. there is always an agent with the required capability for performing a task. Therefore, all tasks will be completed eventually. It was shown with statistical relevance[3], considering data from 30 runs for each unique experiment, that using the GLocal approach shortens the execution times in presence of failures, as a result of the minimisation of calls to the centralised global planner.

C4: Application of the willingness to interact formalism in hunting mobile search.

The fourth contribution of this thesis builds upon contribution C2 and is problem-oriented, i.e. it targets the application of the willingness to interact abstraction for solving the online multi-object κ-coverage problem from the hunting mobile search domain (Paper *D*). The proposed approach is compared with state-of-art methods [77].

The online multi-object κ-coverage problem[4] consists in having at least κ agents covering every target in a given set. Several assumptions and limitations hold. Agents have no initial knowledge regarding neither the whereabouts of the targets, nor their number. Agents and targets are able to move within a bounded 2D space Z, which has a known width and height. Both can move with non-constant, yet limited velocity. Furthermore, agents have higher velocities than any target. The size of each group is constant, thus no agent or target can appear/disappear in Z. Agents have a visibility range within which they

[3] Given that the sample data has no normal distribution, i.e. data can be highly skewed and can have extended tail, the median value was chosen over the mean value and consequently the non-parametric Wilcoxon rank sum test was used.

[4] To be referred to from hereon as the κ-coverage problem for the sake of simplicity.

are able to detect and observe any target. Targets have a constant interest level perceived in the same way by all agents.

In order to solve the κ-coverage problem, agents need to continuously explore the area Z, and upon discovering a target create coalitions with κ members tasked with following and covering such target. The creation of coalitions means that agents need to cooperate with one another for every target that comes along. Therefore, such problem was considered suitable for the willingness to interact abstraction, and was tackled in Paper *D*. The proposed solution consists of three parts. The first part concerns the agent architecture, which was adapted from Paper *B* to display behaviours needed for solving the κ-coverage problem, i.e. explore area and follow targets. The second part focuses on the willingness to interact. An important distinction is made in Paper *D* (relevant also for Paper *E*) between the general disposition of an agent to cooperate with others, and the disposition to cooperate over a specific task. The former depends on the internal state of an agent, such as the battery level and the level of activity of an agent (factors considered in Paper *D*). The latter depends on the task at hand, specifically on the utility of an agent for completing such tasks. In Paper *D* utility is impacted by the level of interest of targets. Thus, the willingness to interact for a task is an aggregation of the factors reflecting the internal state of an agent, and the utility for doing the task. The third part of the solution consists of the interaction protocols put in place for the creation of coalitions and the triggers that activate them. In Paper *D* four such triggers are used: (i) a new target is spotted, (ii) a covered target is moving away from the visibility range of an agent, (iii) an agent is trying to extend an existing coalition to κ members, and (iv) an agent needs to ask for help[5]. The first three triggers are dependent on the application domain, however the fourth one is inherent to the agent itself.

The proposed approach was evaluated in simulation against six methods from the state-of-the-art [77], based on the Robot Operating System (ROS) middle-ware. Two scenarios were investigated, with static and dynamic targets respectively. The performance metrics differ between scenarios because when targets are static, after discovery they remain covered for the rest of the simulation. When targets are mobile such assumption does not hold. Therefore, for the static scenario two metrics are considered: (i) average time to cover one target with at least κ agents and (ii) minimum time to cover all the targets with at least κ agents. Whereas for the dynamic scenario two other metrics are used: (i) average time for which a target is covered with at least κ agents and (ii) aver-

[5]Note that the willingness to interact takes values in $[-1,1]$. A negative willingness denotes that an agent needs to ask for help, while a positive willingness denotes that an agent can give help.

age number of agents that cover one target. In each scenario simulations were run for different numbers of targets $n_t \in \{1,4,7,10,13,16,19\}$. It was shown that, for the static scenario, the willingness to interact approach performs comparably well on average with respect to the rest. With respect to the dynamic scenario, the method with the willingness is overall on the Pareto frontier, and optimises the average number of agents covering a target, on average.

C5: Application of the willingness to interact formalism in mobile wireless sensor networks.
The fifth contribution of this thesis also builds upon contribution C2 and is problem-oriented, i.e. it targets the application of the willingness to interact abstraction for extending the coverage and longevity of mobile wireless sensor networks. The proposed approach is compared with a state-of-art method [78].

In MWSN the problem to be solved relates to finding strategies for the positioning of nodes (agents) in the network – once nodes start depleting and the initial deployment is no longer viable – such that the life of the network is extended as much as possible, and the coverage of the area of interest is maximised. The following assumptions hold. The area of interest, Z, is known with defined width and height. The number of agents/robots, that are to cover Z and maintain network connectivity, is given. The longevity of an agent depends on its battery level, which in turn depends on how much an agent moves, and how much traffic it routes.

In order to address this problem, a hybrid agent approach was proposed (Paper E), which combines the agent approach based on the willingness to interact, with a known reactive approach, namely the Social Potential Fields algorithm (SPF) [78, 79]. This method consists of two phases – SPF and agent negotiations – that alternate with each other during the lifetime of the network. During SPF the network balances such that connectivity is not lost and agents are not too close with each other. This is achieved by specifying several forces that guide the motion of an agent. In this work three such forces were considered: (i) the repulsion force, intended to keep robots from hitting obstacles, (ii) the deployment repulsion force, intended to have robots spread to farther parts of Z, and (iii) the cohesion force, attraction toward the access point (AP), intended to keep robots within a distance of each other in order to avoid loss of communication. At the end of an SPF round nodes do not move any longer and keep routing traffic. As time passes, nodes start depleting their battery levels. Some nodes might deplete faster than others, as such it is of interest to have the nodes with higher battery levels relocate to parts of the network with higher traffic. The willingness to interact is applied here to determine

the cooperative behaviour of the agents, and it depends solely on the battery level, whereas the utility for going to a new position further refines the willingness (similarly to C4). In order to evaluate the willingness the following thresholds for the battery level are introduced: (i) b_{l0} the amount of battery an agent needs to return to a collection point, namely the critical battery level; (ii) b_{lh} given as $10\% \cdot b_{l0} + b_{l0}$, namely the hysteresis battery level; (iii) b_{l1} given as $30\% \cdot b_{l0} + b_{l0}$, namely the level above which the robot is assumed to have enough amount of battery left. This means that, a robot below b_{l1} is in need of help, as opposed to an agent above b_{l1} which is able to give help. Moreover, an agent below b_{l0} should drop its position and return to the collection point, since it only has enough battery for covering that distance. The negotiation protocol is triggered when at least one agent's battery goes below the critical level. Thereafter, all agents with negative willingness will ask for help, with all the requests coming from agents with willingness below the hysteresis level being regarded, as well as the ones with willingness no more than 20% over the hysteresis level. The hysteresis level is used to identify agents which are rather close to the critical level and allowing the network to adapt accordingly, thus avoiding the activation of SPF every time a single agent reaches the critical level. When agents with positive willingness get a request for help, they calculate the utility of going to the requested position. Two factors are considered in the calculation: (i) the distance to the new position and the distance from the new position to the collection point, and (ii) the routed traffic at the new position. The former will determine if the agent will take part in the negotiation at all, i.e. if an agent does not have at least 30% of battery additionally to what is required for the motion, then it will not interact further. Otherwise it will increase/decrease the willingness for a lower/higher traffic load at the the requested position.

The proposed approach was evaluated in simulation and compared against the solution with only SPF. The impact of the cohesion force in each case was also investigated by considering two different values α_1, α_2, where $\alpha_1 < \alpha_2$. A total of four scenarios were considered, two with SPF with α_1, α_2 respectively, and two with the hybrid approach with α_1, α_2 respectively. It was shown that, the hybrid method with the lower cohesion force outperforms the rest with respect to the coverage, longevity, and uniformity of the network. Indeed in the combined approach the cohesion force does not have as much of a central role because the nodes in the outskirts of the network – where the traffic is low – tend to go toward the centre of the network, where the traffic is high. Furthermore, the messages exchanged due to the negotiations in the network are accounted for, and their impact has been shown to be negligible. Note that a

Table 3.1: Mapping between research questions and contributions

	C1	C2	C3	C4	C5
RP1	x				
RP2		x			
RP3			x		
RP4				x	x

limited flooding strategy is assumed, where each message sent is re-transmitted by a node if the latter is located closer to the destination of the message.

The mapping between research problems and contributions is given in Table 3.1.

3.3 Research Process

The research process followed for conducting the research described in this thesis is structured as in Figure 3.1. Initially, a research problem is identified, followed by a review of the literature which highlights the gaps where it is possible to make a contribution. Potentially, the research problem could be refined to accommodate for the knowledge gained during the review. Thereafter, a hypothesis is formulated, along with several research questions, which narrow down the scope of the research problem. In order to tackle the research questions, a solution(s) is proposed which is implemented, and finally evaluated through a scientific method. Depending on the outcome, the proposed solutions can be modified to account for issues that were not foreseen initially,

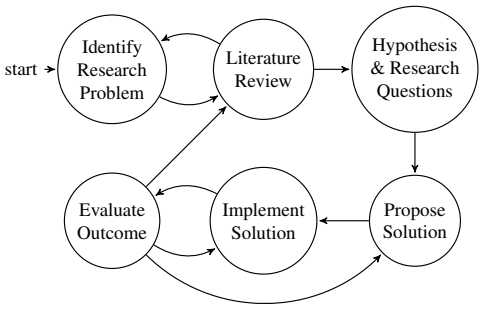


Figure 3.1: The research process

or a problem in the implementation could be identified. Otherwise, the process goes back to the literature review, and proceeds to the next round.

Research methods in computer science have been subject of several studies which have in turn proposed road-maps and guidelines for conducting research in the field [80–82]. Dodig-Crnković has identified three research methods that can be adopted in a research project in computer science: (i) classical methods and processes for building theories based on logics and mathematics adopted in theoretical computer science, (ii) prototyping as an approach for solving problems used in experimental computer science, and (iii) computer simulations, generally used for the investigation of phenomena that either cannot or are too costly to replicate in laboratories [83]. In this thesis, in papers Papers *B-E* the evaluation of the proposed approach is done through computer simulations. Whereas, in Paper *A* taxonomies for MAS properties and MRTA problems were identified and used as a basis for the proposed model.

There is a close relation between the computer simulation and MAS fields, such that simulations can serve as tools in MAS research and vice-versa [84]. Concerning the former view, in the words of Shannon [85]: "The process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behaviour of the system and/or of evaluating various strategies (within the limits imposed by a criterion or a set of criteria) for the operation of the system." Classic examples of the use of simulation as a computational tool for the evaluation of MAS are: the Contract Net Protocol (CNET) [86], the Distributed Monitoring Vehicle Testbed (DMVT) [87], and MACE [88].

There are three key elements that make up a MAS [89], i.e. the agent, the environment[6], and the interactions between agents themselves and with the environment. In Papers *B-D* these three components were implemented on the Robot Operating System (ROS) middle-ware. Every agent is a separate process, or node in ROS terminology, and was modelled as a state-machine with states covering behaviours such as idling, executing, and interacting, which were specialised depending on the needs of each paper, e.g. in Paper *D* the execute state consists of following a set of targets in the 2D space. Agents communicate with one another with message passing using the broadcast, service, and action server mechanisms available in ROS. The environment is a separate node, which generates the 2D space, the tasks, such as emergency sites, or targets, and runs the evolution functions for these tasks, e.g. the mobility of targets. Furthermore, it keeps track of the location of all agents, and

[6]Note the difference between the environment as a model of the real world, and the software infrastructure upon which the MAS is run. In this thesis, environment refers to the former concept.

provides a locking mechanism (Papers *C* and *D*), such that no two agents initiate a collaboration for one task at the same time. The communication between agents and environment is performed with the ROS communication mechanisms. Nodes run concurrently. On the other hand, the agent approach in Paper *E* was evaluated in the simulator used for the evaluation of SPF [90], proposed in previous work by the same authors [79]. This simulator is based on the Player/Stage platform [91], covering both the simulation of agents and of the 2D space within which agents move. There is no concurrency in this model.

In simulation, two properties are desired: replicability and reproducibility. In regards to the former issue, the simulators and simulation code used in Papers *B*-*E* have been made publicly available. Whereas reproducibility of MAS simulations is dependent on the modelling of time, i.e. how agents and environments evolve from one moment in time to another. Three main approaches are discussed in the literature: continuous time, discrete time, and discrete event-based [84]. In Paper *B* there is no explicit modelling of time, i.e. the system time is used. A more structured approach was taken for Papers *C* and *D*, where time was modelled discretely. To this end, a clock node was implemented, which would tick after all the agents and environment had updated – run their evolution functions – once. The communication with the clock is done through the ROS communication mechanisms. In Paper *E* a slightly different approach was taken. The simulation runs iterations until at least 30% of the nodes are operating. One iteration consists of sequential updates of the states of the agents and environment. When SPF is simulated alone [90], at the end of every iteration, the simulation is forwarded to that point in time where at least one agent goes out of order as a result of low battery (triggering the execution of SPF). In the combined approach the simulation is slowed down, i.e. the intermediate steps between the stabilisation of the network – nodes are stationary – and the moment the first agents go out of order are of interest. In that intermission agents ask for help, and negotiate with one another.

Chapter 4

Overview of the Included Papers

Five papers are included in the thesis, referred to as paper *A*-*E*.

4.1 Paper A: TAMER: Task Allocation in Multi-robot Systems Through an Entity-Relationship Model

Abstract – Multi-robot task allocation (MRTA) problems have been studied extensively in the past decades. As a result, several classifications have been proposed in the literature targeting different aspects of MRTA, with often a few commonalities between them. The goal of this paper is twofold. First, a comprehensive overview of early work on existing MRTA taxonomies is provided, focusing on their differences and similarities. Second, the MRTA problem is modelled using an Entity-Relationship (ER) conceptual formalism to provide a structured representation of the most relevant aspects, including the ones proposed within previous taxonomies. Such representation has the advantage of (i) representing MRTA problems in a systematic way, (ii) providing a formalism that can be easily transformed into a software infrastructure, and (iii) setting the baseline for the definition of knowledge bases, that can be used for automated reasoning in MRTA problems.

Authors: Branko Miloradović, Mirgita Frasheri, Baran Çürüklü, Mikael Ek-

ström, and Alessandro V. Papadopoulos.
Status: Published at the 22nd International Conference on Principles and Practice of Multi-Agent Systems (PRIMA'19).
My contribution: With the first co-author we contributed equally to the development of the idea, and the writing process of the paper. The other co-authors provided comments and feedback throughout all the stages, and supported the writing process.

Addressed RPs: RP1.

4.2 Paper B: Adaptive Autonomy in a Search and Rescue Scenario

Abstract – Adaptive autonomy plays a major role in the design of multi-robots and multi-agent systems, where the need of collaboration for achieving a common goal is of primary importance. In particular, adaptation becomes necessary to deal with dynamic environments, and scarce available resources.

In this paper, a mathematical framework for modelling the agents' willingness to interact and collaborate, and a dynamic adaptation strategy for controlling the agents' behaviour, which accounts for factors such as progress toward a goal and available resources for completing a task among others, are proposed. The performance of the proposed strategy is evaluated through a fire rescue scenario, where a team of simulated mobile robots need to extinguish all the detected fires and save the individuals at risk, while having limited resources. The simulations are implemented as a ROS-based multi agent system, and results show that the proposed adaptation strategy provides a more stable performance than a static collaboration policy.

Authors: Mirgita Frasheri, Baran Çürüklü, Mikael Ekström, and Alessandro V. Papadopoulos.
Status: Published at the 12th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO'18).
My contribution: I was the main driver of the paper and contributed with the idea, implementation, and writing. My co-authors provided feedback throughout all the stages.

Addressed RPs: RP2.

4.3 Paper C: GLocal: A Hybrid Approach to the Multi-Agent Mission Re-Planning Problem

Abstract – Multi-robot systems can be prone to failures during plan execution, depending on the harshness of the environment they are deployed in. As a consequence, initially devised plans may no longer be feasible, and a re-planning process needs to take place to re-allocate any pending tasks. Two main approaches emerge as possible solutions, a global re-planning technique using a centralised planner that will redo the task allocation with the updated world state information, or a decentralised approach that will focus on the local plan reparation, i.e. the re-allocation of those tasks initially assigned to the failed robots. The former approach produces an overall better solution, while the latter is less computationally expensive. The goal of this paper is to exploit the benefits of both approaches, while minimising their drawbacks. To this end, we propose a hybrid approach that combines a centralised planner with decentralised multi-agent planning. In case of an agent failure, the local plan reparation algorithm tries to repair the plan through agent negotiation. If it fails to re-allocate all of the pending tasks, the global re-planning algorithm is invoked, which re-allocates all unfinished tasks from all agents. The hybrid approach was compared to planner approach, and it was shown that it improves on the make-span of a mission in presence of different numbers of failures, as a consequence of the local plan reparation algorithm.

Authors: Mirgita Frasheri, Branko Miloradović, Baran Çürüklü, Mikael Ekström, and Alessandro V. Papadopoulos.
Status: Submitted at the 1st International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS'20).
My contribution: The work done on this paper is equally split between Branko Miloradović and I. I developed and implemented the autonomous agents in Python and ROS, while he focused on the development and implementation of the planner in C++. We both worked on the integration part of the work. We equally contributed to the paper's idea, mission scenario creation, and analysis of the obtained results. All co-authors contributed by discussions, feedback, and reviewing the paper.

Addressed RPs: RP3.

4.4 Paper D: Modelling the Willingness to Interact in Cooperative Multi-Robot Systems

Abstract – When multiple robots are required to collaborate in order to accomplish a specific task, they need to be coordinated in order to operate efficiently. To allow for scalability and robustness, we propose a novel distributed approach performed by autonomous robots based on their willingness to interact with each other. This willingness, based on their individual state, is used to inform a decision process of whether or not to interact with other robots within the environment. We study this new mechanism to form coalitions in the online multi-object κ-coverage problem, and compare it with six other methods from the literature. We investigate the trade-off between the number of robots available and the number of potential targets in the environment. We show that the proposed method is able to provide comparable performance to the best method in the case of static targets, and to achieve a higher level of coverage with respect to the other methods in the case of mobile targets.

Authors: Mirgita Frasheri, Lukas Esterle, and Alessandro V. Papadopoulos.
Status: Published at the 12th International Conference on Agents and Artificial Intelligence (ICAART'20).
My contribution: I was the main driver of the paper and contributed with the idea, implementation, and writing. My co-authors provided comments and feedback throughout all the stages.

Addressed RPs: RP4.

4.5 Paper E: Adaptive Autonomy in Wireless Sensor Networks

Abstract – Moving nodes in a Mobile Wireless Sensor Network (MWSN) typically have two maintenance objectives: (i) extend the coverage of the network as long as possible to a target area, and (ii) extend the longevity of the network as much as possible. As nodes move and also route traffic in the network, their battery levels deplete differently for each node. Dead nodes lead to loss of connectivity and even to disengaging full parts of the network. Several reactive and rule-based approaches have been proposed to solve this issue by adapting redeployment to depleted nodes. However, in large networks a deliberative approach may increase performance by taking the evolution of node battery and

traffic into account. In this paper, we present a hybrid agent-based architecture that addresses the problem of depleting nodes during the maintenance phase of a MWSN. Agents, each assigned to a node, collaborate and adapt their behaviour to their battery levels. The collaborative behaviour is modelled through the willingness to interact abstraction, which defines when agents ask and give help to one another. Thus, depleting nodes may ask to be replaced by healthier counterparts and move to areas with less traffic or to a collection point. At the lower level, negotiations trigger a reactive navigation behaviour based on Social Potential Fields (SPF). Results show that the proposed method improves coverage and extends the longevity of the network in an environment without obstacles, by 50% and 13 days, respectively.

Authors: Mirgita Frasheri, Jose Cano-Garcia, Eva Gonzalez-Parada, Baran Çürüklü, Mikael Ekström, Alessandro V. Papadopoulos, and Cristina Urdiales.
Status: Published at the 19th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'20).
My contribution: I was the main driver of the paper and contributed with the idea, implementation, and writing. The second co-author helped with the implementation, writing of sections related to wireless sensor networks (together with the third and last co-author), and provided feedback throughout all the stages. The other co-authors provided comments and feedback throughout all the stages.

Addressed RPs: RP4.

4.6 Other Papers

Other papers not included in the thesis are:

- "Test Agents: The Next Generation of Test Cases". Eduard Paul Enoiu, Mirgita Frasheri. 2nd IEEE Workshop on NEXt level of Test Automation (NEXTA 2019).
- "Analysis of Perceived Helpfulness in Adaptive Autonomous Agent Populations". Mirgita Frasheri, Baran Çürüklü, Mikael Ekström. LNCS Transactions on Computational Collective Intelligence (LNCS TCCI 2018). Invited journal.
- "Comparison Between Static and Dynamic Willingness to Interact in Adaptive Autonomous Agents". Mirgita Frasheri, Baran Çürüklü, Mikael

Ekström. 10th International Conference on Agents and Artificial Intelligence (ICAART'18).

- "Algorithms for the Detection of First Bottom Returns and Objects in the Water Column in Side-Scan Sonar Images". Mohammed Al-Rawi, Fredrik Elmgren, Mirgita Frasheri, Baran Çürüklü, Xin Yuan, José-Fernán Martínez-Ortega, Joaquim Bastos, Jonathan Rodriguez, Marc Pinto. OCEANS'17 conference at the AECC.
- "An Optimized, Data Distribution Service-Based Solution for Reliable Data Exchange Among Autonomous Underwater Vehicles". Jesús Rodríguez Molina, Sonia Bilbao, Belén Martínez, Mirgita Frasheri, and Baran Çürüklü. Sensors 17, no. 8 (2017): 1802.
- "Failure Analysis for Adaptive Autonomous Agents using Petri Nets". Mirgita Frasheri, Lan Anh Trinh, Baran Çürüklü, Mikael Ekström. 11th International Workshop on Multi-Agent Systems and Simulation (MAS&S'17).
- "Towards Collaborative Adaptive Autonomous Agents". Mirgita Frasheri, Baran Çürüklü, Mikael Ekström. 9th International Conference on Agents and Artificial Intelligence 2017 (ICAART'17).

Chapter 5

Conclusion

In this thesis, a formalism for modelling adaptive autonomous behaviour has been proposed, which underlies collaboration between agents in a MAS. The key concept of the formalism is the willingness to interact, composed of the willingness to give and ask for help. A mathematical framework has been designed which can be used for the evaluation of the willingness based on the factors that should have an impact in the decision-making of an agent, and their respective weights. The factors adopted in the included papers do not represent an exhaustive list. Indeed their selection depends on what is considered relevant in a specific application domain.

The willingness to interact formalism has been adopted in solving two problems such as (i) the κ-coverage from the hunting mobile search domain, and (ii) the coverage and network longevity problems from the mobile wireless sensor network domain. In the former case, the approach with the willingness was compared against six state-of-art methods, and improved the κ-coverage metric in scenarios with mobile targets, as well as overall residing in the Pareto frontier. In the latter case, the willingness formalism was combined with a reactive algorithm, namely the social potential fields (SPF), and compared against a purely SPF approach. Results have shown how the combined approach outperforms SPF, in terms of coverage and uniformity of the network, while considering its longevity.

The willingness to interact formalism is purely distributed, i.e. each agent computes its disposition to collaborate based on its local information. While distributed approaches can provide flexibility in terms of graceful degradation of the system in the presence of failures, centralised approaches can provide

optimal solutions in terms of task allocation. This thesis further contributes by integrating these two paradigms, specifically the willingness to interact and centralised high-level planning, in order to exploit the advantages of both. This hybrid approach produced lower mission make-spans, and reduced calls to the centralised planner, as compared to the purely centralised approach.

Conclusively, the work described in this thesis tackles the problem of realising adaptive autonomous agents from several perspectives, i.e. by considering only the collaboration between agents, by considering the collaboration of agents with centralised approaches, and finally by applying the collaboration formalism for solving concrete problems and as a result directly comparing with other relevant work.

Chapter 6

Future Work

There are several research directions of interest that could be investigated in future work such as:

Trust in MAS: The concept of trust can be viewed either in the context of a human trusting the execution of an autonomous system and the information it provides, or in the context of an autonomous systems trusting the execution and information provided by another autonomous system. Regarding the first point of view, Winikoff has proposed three necessary prerequisites for achieving the trust of a human with respect to an autonomous system [92], such as the provision of a framework that allows for alternative courses of action in case the autonomous system behaves in a negative way, ability of the autonomous system to explain itself, and verification and validation of the system, e.g. formal verification. It is noted that each of these prerequisites comes with its own challenges, making the issue of trust a multi-faceted problem. Concerning the second point of view, due to the fact that agents might come from different vendors and are designed with different purposes in mind, benevolence cannot always be assumed. Therefore, it is crucial to investigate how the trust layer can be captured in the formalisms modelling collaborative behaviour between agents, to ensure that benevolent agents are not cheated or lied to during their execution.

Human-in-the-Loop: Agents and robots are envisioned to collaborate in teams also consisting of humans, either as operators, users, or by-standers.

Therefore, the interaction between humans and agents/robots needs to be explicitly addressed, in order to provide models that enable smooth cooperation between them. Such cooperation, and transfers of control between agents and humans are the subject of study of Adjustable Autonomy, that allows the human operator to define the appropriate level of autonomy for an agent. Several issues challenges arise in this context, such as the minimisation of operator workload, and its relation to the levels of autonomy; and the reduction of the opacity of a situation from the point of view of the human operator, that comes as result of loss of situation awareness and common ground between different parties. Finally, an adaptive autonomy approach could be combined with adjustable autonomy, and shift into what is called mixed-initiative interaction, allowing both human and machine to adapt autonomy levels.

Safety and Security: Collaboration between autonomous entities raises safety and security issues that need to be addressed, if such systems are to be integrated in everyday life. Such concerns come as a result of enabling agents to interact with humans, as well as to adapt to what is happening around them, e.g. by relying on information coming from others. Furthermore, mobile agents pose additional threats, as their mobility creates opportunity for spreading potential false or misleading information fast. Key challenges remain: authentication, authorisation, integrity, availability, and confidentiality [93]. Consequently, it is necessary to investigate how such issues can be tackled during the design and implementation of adaptive autonomous systems, with and without the human in the loop.

Industry 4.0: The emergence of the fourth industrial revolution, Industry4.0 (I4.0), has brought forward new problems in which MAS-based approaches could be applied. Industry4.0 started as a national initiative to prepare the German manufacturing industry for the future of production systems, and turned quite quickly into a global initiative intended to affect not only the internal operation of factories, but the development and maintenance of smart future societies and cities [94]. In order to realise this vision, entities, e.g. cyber-physical systems (CPSs), will be connected to one another through enabling technologies such as the internet of things (IoT) and the internet of services (IoS), where they will continuously communicate and cooperate, e.g. in the context of a smart factory, in order to deliver highly customised products and adapt to customer needs, while shortening the time-to-market of these products. An agent in a MAS, as defined in 2, and a CPS are quite close conceptually. Indeed, a CPS is a system composed of (i) hardware with which it

can sense/actuate in its environment, and (ii) software with which it processes the incoming data from the sensors and controls its actuators in order to fulfil specific goals. From an agent perspective, autonomy is rather central to the definition, i.e. the ability to make decisions and act autonomously from external entities – other agents or humans. Within the context of I4.0, CPSs are expected to interact with one another in a dynamic way, make decisions, adapt to changes in the environment or system configurations. Therefore, the presence of some level of autonomy is implied. Furthermore, it has been observed that, challenges for CPSs and CPPSs (cyber-physical production systems), relevant from the perspective of this thesis, lie at the intersection between distributed computing, distributed problem-solving, autonomous system components, and network-based system, and have a close relation to MAS, conventional optimisation techniques, cloud abstractions, and semi heterarchical and heterarchical systems — where the organisation between entities is dynamic [95, 96]. In this context, there is a need for mechanisms that can enable systems to collaborate with one another, when the need arises, in order to fulfil the desired tasks and goals.

Bibliography

[1] Jing Xie and Chen-Ching Liu. Multi-agent systems and their applications. *Journal of International Council on Electrical Engineering*, 7(1):188–197, 2017.

[2] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[3] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade–a fipa-compliant agent framework. In *Proceedings of PAAM*, volume 99, page 33. London, 1999.

[4] Fabio Bellifemine, Giovanni Caire, Agostino Poggi, and Giovanni Rimassa. Jade: A software framework for developing multi-agent applications. lessons learned. *Information and Software Technology*, 50:10–21, 2008.

[5] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[6] Benjamin Hardin and Michael A Goodrich. 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*, pages 165–172. ACM, 2009.

[7] Matthew Johnson, Jeffrey M Bradshaw, Paul J Feltovich, Catholijn M Jonker, M Birna Van Riemsdijk, and Maarten Sierhuis. Coactive design: Designing support for interdependence in joint activity. *Journal of Human-Robot Interaction, 3 (1), 2014*, 2014.

[8] Paul Scerri, David V Pynadath, and Milind Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17(1):171–228, 2002.

[9] Sharon L Poczter and Luka M Jankovic. The google car: driving toward a better future? *Journal of Business Case Studies (Online)*, 10(1):7, 2014.

[10] Murat Dikmen and Catherine M Burns. Autonomous driving in the real world: Experiences with tesla autopilot and summon. In *Proceedings of the 8th international conference on automotive user interfaces and interactive vehicular applications*, pages 225–228, 2016.

[11] Cristiano Castelfranchi. Founding agent's 'autonomy' on dependence theory. In *Proceedings of the 14th European Conference on Artificial Intelligence*, pages 353–357. IOS Press, 2000.

[12] Matthew Johnson, Jeffrey M Bradshaw, Paul J Feltovich, Catholijn M Jonker, Birna Van Riemsdijk, and Maarten Sierhuis. The fundamental principle of coactive design: Interdependence must shape autonomy. In *Coordination, organizations, institutions, and norms in agent systems VI*, pages 172–191. Springer, 2011.

[13] Jeffrey M Bradshaw, Maarten Sierhuis, Alessandro Acquisti, Paul Feltovich, Robert Hoffman, Renia Jeffers, Debbie Prescott, Niranjan Suri, Andrzej Uszok, and Ron Van Hoof. Adjustable autonomy and human-agent teamwork in practice: An interim report on space applications. In *Agent autonomy*, pages 243–280. Springer, 2003.

[14] G Klien, David D Woods, Jeffrey M Bradshaw, Robert R Hoffman, and Paul J Feltovich. Ten challenges for making automation a" team player" in joint human-agent activity. *IEEE Intelligent Systems*, 19(6):91–95, 2004.

[15] Michael A Goodrich, Alan C Schultz, et al. Human–robot interaction: a survey. *Foundations and Trends® in Human–Computer Interaction*, 1(3):203–275, 2008.

[16] Alex John London and David Danks. Regulating autonomous vehicles: A policy proposal. In *AAAI/ACM Conf. AI, Ethics, and Society*, pages 216–221, 2018.

[17] A. Thekkilakattil and G. Dodig-Crnkovic. Ethics aspects of embedded and cyber-physical systems. In *IEEE Computer Software and Applications Conference*, 2015.

[18] Nicolas Cointe, Grégory Bonnet, and Olivier Boissier. Ethical judgment of agents' behaviors in multi-agent systems. In *Autonomous Agents & Multiagent Systems*, pages 1106–1114, 2016.

[19] Aline Belloni, Alain Berger, Olivier Boissier, Grégory Bonnet, Gauvain Bourgne, Pierre-Antoine Chardel, Jean-Pierre Cotton, Nicolas Evreux, Jean-Gabriel Ganascia, Philippe Jaillon, et al. Dealing with ethical conflicts in autonomous agents and multi-agent systems. In *Int. Workshop on Artificial Intelligence and Ethics*, 2015.

[20] Christina Rödel, Susanne Stadler, Alexander Meschtscherjakov, and Manfred Tscheligi. Towards autonomous cars: the effect of autonomy levels on acceptance and user experience. In *Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 1–8. ACM, 2014.

[21] Michael Wooldridge. Intelligent agents: The key concepts. In *ECCAI Advanced Course on Artificial Intelligence*, pages 3–43. Springer, 2001.

[22] Nicholas R Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1(1):7–38, 1998.

[23] Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152, 1995.

[24] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[25] Michael Wooldridge and Nicholas R Jennings. Agent theories, architectures, and languages: a survey. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 1–39. Springer, 1994.

[26] Allen Newel and Herbert A Simon. Computer science as empirical inquiry: Symbols and search. *Communications*, 1976.

[27] David Vernon. *Artificial cognitive systems: A primer*. MIT Press, 2014.

[28] Rodney A Brooks. Elephants don't play chess. *Robotics and autonomous systems*, 6(1-2):3–15, 1990.

[29] Rodney A Brooks. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159, 1991.

[30] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23, 1986.

[31] Innes A Ferguson. Touring machines: Autonomous agents with attitudes. *Computer*, 25(5):51–55, 1992.

[32] David Vernon, Giorgio Metta, and Giulio Sandini. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *IEEE transactions on evolutionary computation*, 11(2):151–180, 2007.

[33] Paul S Rosenbloom, John Laird, and Allen Newell. The soar papers: Research on integrated intelligence. 1993.

[34] John R Anderson, Daniel Bothell, Michael D Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. *Psychological review*, 111(4):1036, 2004.

[35] Juyang Weng. Developmental robotics: Theory and experiments. *International Journal of Humanoid Robotics*, 1(02):199–236, 2004.

[36] Jeffrey L Krichmar and George N Reeke. The darwin brain-based automata: Synthetic neural models and real-world devices. In *Modeling in the Neurosciences*, pages 631–656. CRC Press, 2005.

[37] Ian Horswill. Cerebus: A higher-order behavior-based system. *AI Magazine*, 2002.

[38] Rodney A Brooks, Cynthia Breazeal, Matthew Marjanović, Brian Scassellati, and Matthew M Williamson. The cog project: Building a humanoid robot. In *International Workshop on Computation for Metaphors, Analogy, and Agents*, pages 52–87. Springer, 1998.

[39] Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. *Self-organising Software*. Springer, 2011.

[40] Gerhard Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.

[41] Edmund H Durfee and Jeffrey S Rosenschein. Distributed problem solving and multi-agent systems: Comparisons and examples. In *Proceedings of the Thirteenth International Distributed Artificial Intelligence Workshop*, pages 94–104, 1994.

[42] Edmund H Durfee, Victor R Lesser, and Daniel D Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 100(11):1275–1291, 1987.

[43] Katia P Sycara. Multiagent systems. *AI magazine*, 19(2):79–79, 1998.

[44] Jacques Ferber and Gerhard Weiss. *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading, 1999.

[45] Jeffrey M Bradshaw, Hyuckchul Jung, Shri Kulkarni, Matthew Johnson, Paul Feltovich, James Allen, Larry Bunch, Nathanael Chambers, Lucian Galescu, Renia Jeffers, et al. Toward trustworthy adjustable autonomy in kaos. In *Trusting Agents for Trusting Electronic Societies*, pages 18–42. Springer, 2005.

[46] Thomas B Sheridan and William L Verplank. Human and computer control of undersea teleoperators. Technical report, Massachusetts Inst of Tech Cambridge Man-Machine Systems Lab, 1978.

[47] Raja Parasuraman, Thomas B Sheridan, and Christopher D Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, 30(3):286–297, 2000.

[48] Jonathan Brookshire, Sanjiv Singh, and Reid Simmons. Preliminary results in sliding autonomy for assembly by coordinated teams. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 706–711. IEEE, 2004.

[49] Tom Ziemke. On the role of emotion in biological and robotic autonomy. *BioSystems*, 91(2):401–408, 2008.

[50] Terrence Fong, Charles Thorpe, and Charles Baur. *Collaborative control: A robot-centric model for vehicle teleoperation*, volume 1. Carnegie Mellon University, The Robotics Institute, 2001.

[51] Michael A Goodrich, Dan R Olsen, Jacob W Crandall, and Thomas J Palmer. Experiments in adjustable autonomy. In *Proceedings of IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents*, pages 1624–1629, 2001.

[52] Paul J Feltovich, Jeffrey M Bradshaw, William J Clancey, and Matthew Johnson. Toward an ontology of regulation: Socially-based support for coordination in human and machine joint activity. In *International Workshop on Engineering Societies in the Agents World*, pages 175–192. Springer, 2006.

[53] Jeffrey M Bradshaw, Hyuckchul Jung, Shri Kulkarni, Matthew Johnson, Paul Feltovich, James Allen, Larry Bunch, Nathanael Chambers, Lucian Galescu, Renia Jeffers, 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*, pages 214–221. ACM, 2005.

[54] Milind Tambe. Agent architectures for flexible. In *Proc. of the 14th National Conf. on AI, USA: AAAI press*, pages 22–28, 1997.

[55] Nathan Schurr, Janusz Marecki, and Milind Tambe. Improving adjustable autonomy strategies for time-critical domains. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 353–360. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[56] Stephen G McGill, Seung-Joon Yi, Hak Yi, Min Sung Ahn, Sanghyun Cho, Kevin Liu, Daniel Sun, Bhoram Lee, Heejin Jeong, Jinwook Huh, et al. Team thor's entry in the darpa robotics challenge finals 2015. *Journal of Field Robotics*, 2016.

[57] John E Laird. *The Soar cognitive architecture*. MIT Press, 2012.

[58] HJ Levesque, PR Cohen, and J Nunes. On acting together. inproceedings of the national conference on artificial intelligence, 1990.

[59] Cheryl Martin and K Suzanne Barber. Adaptive decision-making frameworks for dynamic multi-agent organizational change. *Autonomous Agents and Multi-Agent Systems*, 13(3):391–428, 2006.

[60] Bennie Lewis, Bulent Tastan, and Gita Sukthankar. An adjustable autonomy paradigm for adapting to expert-novice differences. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1656–1662. IEEE, 2013.

[61] Sebastian Muszynski, Jörg Stückler, and Sven Behnke. Adjustable autonomy for mobile teleoperation of personal service robots. In *RO-MAN, 2012 IEEE*, pages 933–940. IEEE, 2012.

[62] Andreas Birk and Max Pfingsthorn. A hmi supporting adjustable autonomy of rescue robots. In *Robot Soccer World Cup*, pages 255–266. Springer, 2005.

[63] Gloria L Calhoun, Michael A Goodrich, John R Dougherty, and Julie A Adams. Human-autonomy collaboration and coordination toward multi-rpa missions. *Remotely Piloted Aircraft Systems: A Human Systems Integration Perspective*, page 109, 2016.

[64] Michael J Barnes, Jessie YC Chen, and Florian Jentsch. Designing for mixed-initiative interactions between human and autonomous systems in complex environments. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, pages 1386–1390. IEEE, 2015.

[65] Kristen Stubbs, Pamela J Hinds, and David Wettergreen. Autonomy and common ground in human-robot interaction: A field study. *IEEE Intelligent Systems*, 22(2), 2007.

[66] B v Vecht, Frank Dignum, and JJ Meyer. Autonomy and coordination: Controlling external influences on decision making. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*, pages 92–95. IEEE Computer Society, 2009.

[67] Sanghyun Kim, Mingon Kim, Jimin Lee, Soonwook Hwang, Joonbo Chae, Beomyeong Park, Hyunbum Cho, Jaehoon Sim, Jaesug Jung, Hosang Lee, et al. Team snu's control strategies for enhancing a robot's capability: Lessons from the 2015 darpa robotics challenge finals. *Journal of Field Robotics*, 2016.

[68] Alessandro Farinelli, Masoume M Raeissi, Nathan Brooks, Paul Scerri, et al. Interacting with team oriented plans in multi-robot systems. *Autonomous Agents and Multi-Agent Systems*, pages 1–30, 2016.

[69] Gregory Dudek, Michael R. M. Jenkin, Evangelos Milios, and David Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397, 1996.

[70] Y. Uny Cao, Alex S. Fukunaga, Andrew B. Kahng, and Fanfan Meng. Cooperative mobile robotics: antecedents and directions. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 1, pages 226–234, 1995.

[71] Brian P. Gerkey and Maja J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *Int. J. Robotics Res.*, 23(9):939–954, 2004.

[72] David Landén, Fredrik Heintz, and Patrick Doherty. Complex task allocation in mixed-initiative delegation: a uav case study. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 288–303. Springer Berlin Heidelberg, 2010.

[73] G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *Int. J. Robotics Res.*, 32(12):1495–1512, 2013.

[74] Ernesto Nunes and Maria Gini. Multi-robot auctions for allocation of tasks with temporal constraints. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2110–2216, 2015.

[75] Cristiano Castelfranchi. Guarantees for autonomy in cognitive agent architecture. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 56–70. Springer, 1994.

[76] Branko Miloradović, Baran Çürüklü, Mikael Ekström, and Alessandro V Papadopoulos. A genetic algorithm approach to multi-agent mission planning problems. In *International Conference on Operations Research and Enterprise Systems*, pages 109–134. Springer, 2019.

[77] Lukas Esterle and Peter R. Lewis. Online multi-object k-coverage with mobile smart cameras. In *Proc. of the Int. Conf. on Distributed Smart Cameras*, pages 1–6. ACM, 2017.

[78] Eva González-Parada, Jose Cano-García, Francisco Aguilera, Francisco Sandoval, and Cristina Urdiales. A social potential fields approach for

self-deployment and self-healing in hierarchical mobile wireless sensor networks. *Sensors*, 17(1):120, 2017.

[79] Cristina Urdiales, Francisco Aguilera, Eva González-Parada, Jose Cano-García, and Francisco Sandoval. Rule-based vs. behavior-based self-deployment for mobile wireless sensor networks. *Sensors*, 16(7):1047, 2016.

[80] Marvin V Zelkowitz and Dolores Wallace. Experimental validation in software engineering. *Information and Software Technology*, 39(11):735–743, 1997.

[81] Mary Shaw. What makes good research in software engineering? *International Journal on Software Tools for Technology Transfer*, 4(1):1–7, 2002.

[82] Hilary J Holz, Anne Applin, Bruria Haberman, Donald Joyce, Helen Purchase, and Catherine Reed. Research methods in computing: what are they, and how should we teach them? In *ACM SIGCSE Bulletin*, volume 38, pages 96–114. ACM, 2006.

[83] Gordana Dodig-Crnkovic. Scientific methods in computer science. In *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden, Skövde, Suecia*, pages 126–130, 2002.

[84] Fabien Michel, Jacques Ferber, and Alexis Drogoul. Multi-agent systems and simulation: A survey from the agent commu-nity's perspective. In *Multi-Agent Systems*, pages 17–66. CRC Press, 2018.

[85] Robert E Shannon. Systems simulation; the art and science. Technical report, 1975.

[86] Reid G Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on computers*, (12):1104–1113, 1980.

[87] Victor R Lesser and Daniel G Corkill. The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks. *AI magazine*, 4(3):15–15, 1983.

[88] Les Gasser. Mace: A flexible testbed for distributed al research. *Distributed artificial intelligence*, 1987.

[89] Yves Demazeau. Steps toward multi-agent programming. *IWMAS-97*, 1997.

[90] Eva González-Parada, Jose Cano-García, Francisco Aguilera, Francisco Sandoval, and Cristina Urdiales. A social potential fields approach for self-deployment and self-healing in hierarchical mobile wireless sensor networks. *Sensors*, 17(1):120, 2017.

[91] Brian Gerkey, Richard T Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th international conference on advanced robotics*, volume 1, pages 317–323, 2003.

[92] Michael Winikoff. Towards trusting autonomous systems. In *International Workshop on Engineering Multi-Agent Systems*, pages 3–20. Springer, 2017.

[93] Ali Dorri, Salil S Kanhere, and Raja Jurdak. Multi-agent systems: A survey. *IEEE Access*, 6:28573–28593, 2018.

[94] Vasja Roblek, Maja Meško, and Alojz Krapež. A Complex View of Industry 4.0. *SAGE Open*, 6(2), 2016.

[95] Luis Ribeiro. Cyber-physical production systems' design challenges. *IEEE International Symposium on Industrial Electronics*, pages 1189–1194, 2017.

[96] Luis Ribeiro and Mats Bjorkman. Transitioning from Standard Automation Solutions to Cyber-Physical Production Systems: An Assessment of Critical Conceptual and Technical Challenges. *IEEE Systems Journal*, 12(4):3816–3827, 2018.

II

Included Papers

Chapter 7

TAMER: Task Allocation in Multi-Robot Systems Through an Entity-Relationship Model

Authors: Branko Miloradović, Mirgita Frasheri, Baran Çürüklü, Mikael Ekström, Alessandro Vittorio Papadopoulos
Venue: 22^{nd} International Conference on Principles and Practice of Multi-Agent Systems (PRIMA'19)

Abstract

Multi-robot task allocation (MRTA) problems have been studied extensively in the past decades. As a result, several classifications have been proposed in the literature targeting different aspects of MRTA, with often a few commonalities between them. The goal of this paper is twofold. First, a comprehensive overview of early work on existing MRTA taxonomies is provided, focusing on their differences and similarities. Second, the MRTA problem is modelled using an Entity-Relationship (ER) conceptual formalism to provide a structured representation of the most relevant aspects, including the ones proposed within previous taxonomies. Such representation has the advantage of (i) representing MRTA problems in a systematic way, (ii) providing a formalism that can be easily transformed into a software infrastructure, and (iii) setting the baseline for the definition of knowledge bases, that can be used for automated reasoning in MRTA problems.

7.1 Introduction

In the past decades, the interest in Multi-Agent Systems (MASs) has grown due to their suitability in representing applications where actors have different interests, and to their distributed nature that increases performance, scalability, and robustness [1]. Earlier papers from the 1980s and 1990s mostly focused on the properties and collaborative behaviour of MASs putting the emphasis on the specific aspects of the problem to be solved, e.g., communication, topology, robot group composition, and collaborative behaviour. Proposed solutions were usually verified in simulation environments.

As the complexity of the MAS missions started to increase, e.g., in terms of number of required agents, number of tasks to be completed, heterogeneity of capabilities required to complete some tasks, etc., more attention has been devoted to the multi-robot task allocation (MRTA) problem, which has become an established research direction [2]. In order to tame such an emerging complexity, several taxonomies have been proposed in the literature. Gerkey and Matarić [3] introduced the first taxonomy for MRTA problems, proposing three main dimensions that specified the type of tasks, type of robots, and type of assignment. Other taxonomies have been proposed in the following years, further highlighting the complexity of the MRTA problem. However, most of them are do not build on previous ones, leading to a fragmented and possibly overlapping set of taxonomies.

This paper surveys the existing taxonomies, in order to capture the important dimensions of MRTA problem configurations and to understand differences and similarities. In addition, this paper presents the Task Allocation in Multi-Robot System Entity-Relationship (TAMER) model, an Entity-Relationship (ER) model that captures the most relevant aspects of the surveyed MRTA taxonomies. The goal of TAMER is to provide a unified view of the existing taxonomies, and a tool to classify and relate the different dimensions in a more structured and systematic way. Adding new dimensions on top of existing taxonomies requires a clear understanding of how they could fit in the big picture. In fact, newly proposed aspects may overlap with, may be coupled with, or may contain certain properties already captured by other dimensions. TAMER simplifies such process providing a more formal approach to tame the complexity of the MRTA taxonomy problem. TAMER offers a general model that includes the different dimensions proposed by the surveyed taxonomies (Section 7.2), and it can be thought of as a unifying approach to the MRTA taxonomy problem, allowing for extending the classification with new dimensions in a non-redundant way, in the attempt of providing a unique

framework for the definition of the relevant dimensions in MRTA problems.

The contribution of this paper is twofold: (i) To provide an overview of MRTA taxonomies, analysing how the research axes evolved over the past few decades, and identifying differences and similarities among them (Section 7.2); (ii) To formalize the MRTA problem through TAMER, an ER conceptual model that includes the most relevant aspects of the identified MRTA research axes (Section 7.3).

7.2 Overview of the MRTA taxonomies

The categorization of the MRTA problems across various dimensions has been extensively investigated by several researchers in the past three decades. Earlier taxonomies [1,4,5], from the 1990s and the beginning of 2000s, focus more on the communication, the cooperation, and the robot capabilities dimensions. Table 9.2 summarizes the main surveyed taxonomies, and the respective proposed dimensions. In these taxonomies, the task allocation dimension plays a minor role. The work presented by Gerkey and Matarić [3] is the first one to shift the focus from former dimensions, into the direction of task allocation. This trend has been followed in the past decade and a half, expanding the original MRTA dimensions [6–8].

The group composition represents a crucial aspect of a MAS, and has been addressed explicitly as the group architecture and size [4], collective composition [5], and degree of heterogeneity [1]. The robot group composition has been addressed in the original MRTA taxonomy with the introduction of Single-Robot (SR) and Multi-Robot (MR) tasks, and Single-Task (ST) and Multi-Task (MT) robots dimensions. In order to have heterogeneity in the robot group composition, individual robots must have different capabilities. The range of robot capabilities is very broad going from the ability to model other agents and learning [4], processing ability [5], to the ability to perform tasks concurrently [3].

The communication and topology dimensions were an important part of early taxonomies, however, with the shift of focus towards task allocation and task interrelatedness, the communication was usually assumed to be failure-free and it did not have an effect on the problem configuration or solution design. Nevertheless, these dimensions are of major importance in MASs and they have been divided into several sub-dimensions. They include the way of interaction [4], the communication range, bandwidth, and topology [5], and the communication language and protocols [1].

Table 7.1: Summary of the proposed dimensions classification in MRTA taxonomies.

Dimension \ Reference	Dudek *et al.* [5]	Cao *et al.* [4]	Stone *et al.* [1]	Lau & Zhang [9]	Gerkey & Matarić [3]	Landén *et al.* [6]	Korsah *et al.* [7]	Nunes *et al.* [8]
Group Composition	✓	✓						
Robot Capabilities	✓	✓	✓		✓			
Communication	✓	✓	✓					
Topology	✓	✓	✓					
Cooperation		✓	✓		✓		✓	
Resources	✓		✓	✓				
Environment		✓				✓		
Allocation				✓	✓	✓		✓
Task Interrelatedness			✓			✓	✓	✓

Another fundamental aspect in MASs is the interaction among agents, which can be intentional or emergent [4]. Furthermore, agents can have competitive or benevolent behaviour, negotiate and make commitments in order to reach their goals [1]. In later papers, the cooperation is usually assumed to be intentional and benevolent [3, 7] or it is not been taken into account at all [6, 8]. When resources are finite [9], resource conflict may arise [4], thus a resource manager is needed [1]. Conflicts can be related to sharing space, objects, equipment, or communication. If agents are physical units acting within an environment, geometric problems may occur [4]. The environment is classically classified as static or dynamic [6]. Sudden and unplanned changes in the environment may have different consequences on the problem configuration, ultimately leading to a task re-allocation.

Another major part of the MRTA taxonomy is the task allocation dimension. This dimension can be further divided into Instantaneous Assignment (IA) and Time-Extended Assignment (TA) [3]. If the allocation is done by an agent, then the allocation is internal and is considered as a task in MAS, otherwise, it is assumed that the allocation process is external [6].

In order to cover the gaps that were left by the taxonomy proposed by Gerkey and Matarić [3], by not addressing interrelated utilities and task constraints, several different taxonomy additions were proposed [6–8]. Landén *et al*. [6] defined unrelated utilities and interrelated utilities as well as independent tasks and constrained tasks. On the other hand, Korsah *et al*. [7] covered both of these dimensions with a single dimension: the degree of interrelatedness. Although not identical, these concepts are related, so both utilities and constraints have an impact on the degree of interrelatedness between both agents and tasks. Instead of utility, Lau and Zhang [9] express the degree of objective fulfilment in profit. Although Gerkey and Matarić [3] state that their work does not include interrelatedness between tasks explicitly, it can be noted that MR tasks do require some sort of synchronization between robots, while MT robots must have intra-related schedules in the case of TA. In addition, Nunes *et al*. [8] distinguish between temporal and ordering constraints, by adding Time Windows (TW) and Synchronization Precedence (SP) under TA. Furthermore, MRTA problem can be deterministic if the output of the model is completely determined by the initial conditions or stochastic if a model of the uncertainty is available. Despite the importance of uncertainty in robotics, most MRTA models are deterministic and deal with uncertainty only at execution time. Finally, all constraints can be divided into hard and soft constraints.

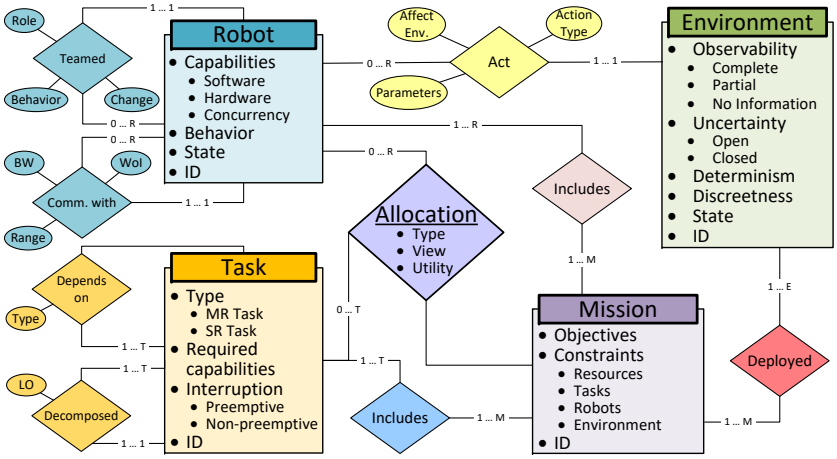


Figure 7.1: The TAMER model.

7.3 The TAMER Model

The TAMER model (shown in Fig. 7.1) aims at covering the relevant aspects of the MRTA problem, by adopting a systematic approach to unify the different dimensions presented in the former taxonomies. TAMER is an Entity-Relationship (ER) model that defines the relevant entities of MRTA, and how they relate among them. TAMER unifies the previously proposed taxonomies, in a unique taxonomy that makes sure that the different dimensions are all necessary and sufficient to describe the fundamental problem configuration. TAMER also includes for all the entities and relationships a minimal set of attributes that captures the most relevant aspects presented in former taxonomies. Note that the proposed set of attributes does not aim for completeness, but it represents a core set that can be easily extended thanks to the TAMER approach.

7.3.1 Entities

TAMER consists of four entities: (i) *Robot*, (ii) *Environment*, (iii) *Task*, and (iv) *Mission*.

Robot. The *Robot* entity consists of the state, behaviour and capability

attributes[1]. The state attribute covers those variables that are considered of interest in a particular context, e.g., velocity, position, orientation, and battery level. Different contexts might require different sets of variables, thus the state attribute is not specified in detail. The behaviour refers to the level of autonomy displayed by a robot. A robot might be able to display a particular level of autonomy that is fixed over time, or the level of its autonomy can be adaptive. Due to changing circumstances, the dependencies among robots can change, and, as a result, the autonomy levels change as well [10]. Both adaptive and fixed autonomy have an impact on the cooperation among the agents. Whereas the former allows for dynamic patterns and different levels of cooperation, the latter implies fixed patterns and a predefined level of cooperation.

The capability attribute covers the abilities of a robot, both at the hardware and software levels. These abilities can correspond to different levels of abstraction. For instance, at a low-level an ability might refer to processing power, concurrency, and/or computational resources, whereas at a high-level an ability might relate to being able of doing some action, e.g., grasping a mug.

Environment. The *Environment* entity is characterized by the following attributes: state, observability, uncertainty, determinism, discreetness, and additional constraints. As for the state attribute, different variables that describe the environment could be relevant in different contexts, e.g., the location of dynamic obstacles at a specific timestamp. The observability attribute takes values such as complete, partial, or no information. The uncertainty, on the other hand, refers to the dynamics in the environment, i.e., whether the environment does not change (closed) or changes overtime (open). Determinism, discreetness are characteristics described by Russell and Norvig [11, Chapter 2]. The additional constraints attribute serve the purposes of describing the environment in terms of rules and laws that are applicable and shape how the problem is formulated.

Task. The task entity consists of type. required capabilities, and interruption attribute. The task type attribute is identical to the Gerkey and Matarić [3] definition of SR and MR tasks. Required capabilities attribute describes the capability a robot needs to possess in order to execute a certain task. If a task can be temporarily interrupted without requiring its cooperation, in order to do some other task, then the task being interrupted is said to be preemptive. Preemptive tasks are of very common occurrence in real-time systems.

Mission. Mission entity encapsulates mission objectives, available resourc-

[1] All entities have an ID attribute, that uniquely distinguishes between instances of the same entity. The ID is not further discussed in this paper.

es, and constraints that are part of the problem domain. This is where the problem configuration as well as the objectives are defined. Mission constraints are constraints, which are imposed by some external actor, which is configuring the mission problem, e.g., human operator. These constraints can relate to resources, robots, tasks, and environment. For example, a constraint, which says that robot *i* can use at most 50% of its battery is considered to be resource constraint. Similarly, a set of *n* tasks to be completed is a task constraint. TW are another example of task constraints. A robot constraint may restrict, e.g., the number of robots that can be used in a specific mission. Specific constraints can be imposed regarding environment, e.g., in the form of forbidden areas, which must not be visited, or crossed.

7.3.2 Relationships

TAMER also includes nine relationships: (i) *Teamed*, (ii) *Communicate with*, (iii) *Act*, (iv) *Depends on*, (v) *Decomposed*, (vi) *Allocation*, (vii) *Includes Robot*, (viii) *Includes Task*, and (ix) *Deployed*.

Teamed. Robots can be part of teams within a MAS, and as such be in a *Teamed* relationship with one another. Attributes that characterize such relationship are state, behaviour, role, and dynamics. The state of a team could be specified by the size of the team, its composition in terms of robot capabilities, and the behaviour of the team. This attribute is similar to the behaviour attribute of the robot entity, however in this case it refers to the overall behaviour of the team that emerges from the local robot behaviours. The role attribute describes what hierarchical position a robot has in a particular team, e.g., leader or peer. The dynamics attribute refers to whether the team can change in time in terms of composition or hierarchy, among other variables.

Communicate with. *Communicate with* is also a relationship between robots, and has four attributes: type, range, bandwidth, and way of interaction. Communication type includes broadcast and one-to-one communication. Range and bandwidth describe physical properties of the communication channel. Way of interaction expresses whether a robot communicates directly with another robot, or indirectly, e.g., stigmergy where communication happens via the environment. The problem can depend on the upper bound of the bandwidth and range, which is a characteristic of a specific environment. Notice that the specification of this relationship defines the network topology, i.e., which robot communicates with whom.

Act. The *Act* relationship connects the robot and the environment entities to each other. A robot can act in an environment, and as a result have an

impact on the state of the environment. Similarly, the environment can act on the robot and affect its state. This relationship is characterized by the type of action, parameters of the action, and affect on environment. A specific action can be described by a set of parameters, e.g., the action name could be one such parameter which defines what the action is. More parameters could be specified depending on the need. The *affect on the environment* attribute distinguishes between active and passive actions on the environment. The former covers actions that change the environment, whereas the latter covers actions that do not change the environment, e.g., a robot's movement.

Depends on. *Depends on* is a relationship between task entities describing their dependencies. This relationship has a *type* attribute. The type attribute, specifies what is the type of task dependency, i.e., *inter-dependent* (there are dependencies within robot's schedule), and *cross-schedule dependent* (there are dependencies within different robots' schedules). These dependencies can be utility related, synchronous, or time windows. Ordering constraints are treated as a special case of synchronization constraints.

Decomposed. Tasks can be atomic or divisible. The representation of the tasks is a design choice, and it may depend on the final purpose of the modeling. Tasks that are considered atomic from a high-level planning perspective, can be seen as divisible at the low level perspective, e.g., when agents need to coordinate to complete a more complex task. For example, Miloradović *et al*. [12] considered MR tasks as atomic in a high-level mission planning approach, while Zlot [13] deal with the task decomposition and allocation with Logical Operators (LO).

Allocation. The main relationship in the taxonomy that binds together mission, task, and robot entity is the allocation. The allocation can assign $0 \dots T$ tasks to $0 \dots R$ robots. If 0 tasks are assigned to 0 robots it means there is no allocation, hence no mission. However, it is still possible to have 0 tasks allocated to m robots, meaning that these m robots will not be used in a mission. The allocation consists of allocation type (IA or TA), allocation view (internal, external [6], or hybrid) and utility function.

Includes. The *includes* relationship connects the mission with the robot and task entities. This defines which tasks and robots are included in the mission. To have a mission, there must be at least 1 task allocated to at least 1 robot.

Deployed. After the allocation is done for a defined mission, through the *deployed* relationship the mission is deployed in the environment for execution. This means that missions are further constrained and shaped by the specific environment they should be executed in.

7.3.3 Discussion

The MRTA problem needs to consider all the presented aspects in order to represent a specific deployment. MRTA algorithms are in charge of populating the allocation relationship, based on the set of available robots, on the mission composed of the different tasks, and on the description of the environment.

The need for the TAMER model is motivated by the emerging complexity, both of the MRTA taxonomies and MAS missions. Most of the proposed taxonomies analyze the MRTA problem from different angles, and possibly introducing additional dimensions that are indirectly covered by other ones. TAMER model has several advantages. First, it allows for a systematic and structured representation of MRTA taxonomies. In fact, the taxonomies presented in Section 7.2 are included or can be reduced to specific instances of the TAMER model, avoiding redundancies and overlaps. For example, different topologies of communication are not directly represented in the TAMER model, but are a result of the relation *Communicate with*, that specifies the adjacency matrix of the communication topology, including additional attributes, such as the Range, the Bandwidth, and the Way of Interaction. Also, in TAMER all the attributes are assumed to be able to vary over time, while keeping a consistent knowledge base of the problem configuration.

The second important advantage of TAMER is that it adopts a classical approach for data/knowledge representation. As a result, TAMER defines a complex data structure that can be used for the definition of software infrastructures in MRTA problems, and for MRTA algorithms. Moreover, the TAMER model can be extended to include additional semantics to enable automated reasoning in MRTA problems.

Finally, TAMER adds two additional research axes: Multi-Mission problems and Multi-Environment problems. It allows multiple missions to be defined and deployed in the multiple or shared environment with the possibility of sharing robots and resources among the missions. The multi-mission and multi-environments aspects have not been extensively explored.

7.4 Conclusion

This work provides an overview of the main taxonomies for MRTA problems, analyzing and relating the different components (in this paper referred to as axes, or dimensions) proposed in the literature. Such dimensions may overlap or represent different aspects of the MRTA problem, but they seldom provide a general view on it. In order to tame the emerging complexity coming from

the different taxonomies, we proposed TAMER, an ER model that provides a unified view on the MRTA problem, with the aim of remove potential redundancies in the classification, as well as a structured way to add or remove additional dimensions. As future work, TAMER can be extended to define a knowledge base for enabling automated reasoning in MRTA problems.

Bibliography

[1] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[2] Prithviraj Dasgupta. *Multi-Robot Task Allocation for Performing Cooperative Foraging Tasks in an Initially Unknown Environment*, pages 5–20. 2011.

[3] Brian P. Gerkey and Maja J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *Int. J. Robotics Res.*, 23(9):939–954, 2004.

[4] Y. Uny Cao, Alex S. Fukunaga, and Andrew B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, 1997.

[5] Gregory Dudek, Michael Jenkin, and Evangelos Milios. A taxonomy of multirobot systems. *Robot teams: From diversity to polymorphism*, pages 3–22, 2002.

[6] David Landén, Fredrik Heintz, and Patrick Doherty. Complex task allocation in mixed-initiative delegation: A uav case study. In *PRIMA*, pages 288–303, 2012.

[7] G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *Int. J. Robotics Res.*, 32(12):1495–1512, 2013.

[8] Ernesto Nunes, Marie Manner, Hakim Mitiche, and Maria Gini. A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics and Autonomous Systems*, 90:55 – 70, 2017.

[9] Hoong Chuin Lau and Lei Zhang. Task allocation via multi-agent coalition formation: taxonomy, algorithms and complexity. In *ICTAI*, pages 346–350, 2003.

[10] Mirgita Frasheri, Baran Curuklu, Mikael Ekström, and Alessandro Vittorio Papadopoulos. Adaptive autonomy in a search and rescue scenario. In *12th IEEE SASO*, 2018.

[11] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[12] Branko Miloradović, Baran Çürüklü, and Mikael Ekström. A genetic mission planner for solving temporal multi-agent problems with concurrent tasks. In *ICSI*, pages 481–493, 2017.

[13] Robert Michael Zlot. *An Auction-Based Approach to Complex Task Allocation for Multirobot Teams*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2006.

Chapter 8

Adaptive Autonomy in a Search and Rescue Scenario

Authors: Mirgita Frasheri, Baran Çürüklü, Mikael Ekström, Alessandro Vittorio Papadopoulos
Venue: 12^{th} IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO'18)

Abstract

Adaptive autonomy plays a major role in the design of multi-robots and multi-agent systems, where the need of collaboration for achieving a common goal is of primary importance. In particular, adaptation becomes necessary to deal with dynamic environments, and scarce available resources. In this paper, a mathematical framework for modelling the agents' willingness to interact and collaborate, and a dynamic adaptation strategy for controlling the agents' behavior, which accounts for factors such as progress toward a goal and available resources for completing a task among others, are proposed. The performance of the proposed strategy is evaluated through a fire rescue scenario, where a team of simulated mobile robots need to extinguish all the detected fires and save the individuals at risk, while having limited resources. The simulations are implemented as a ROS-based multi agent system, and results show that the proposed adaptation strategy provides a more stable performance than a static collaboration policy.

8.1 Introduction

Adaptive autonomy (AA) is the ability of agents to adapt autonomously their behavior to continuously changing circumstances [1]. The implications of this definition are twofold: (i) each agent decides itself on its autonomy levels, and (ii) such decisions are taken continuously during execution. Furthermore, to change autonomy levels means to change the dependence relations among agents [2], e.g., an agent becomes less autonomous if it depends on the assistance of other agents to complete its task.

This paper deals with the development of *local* reasoning mechanisms, adapting the level of cooperation among agents based on several factors in order to improve the overall performance. To this end, AA is modeled herein through the *willingness to interact* [3], which allows agents to decide when to give or ask for help. AA behavior is relevant in those scenarios in which agents should be able to solve issues locally—especially when assistance from human operators is not available, e.g., due to unreliable communication channels. One such application domain is Search and Rescue (SAR). In this paper, a previously developed ROS-based agent simulation [4] has been extended with a SAR scenario, and used to evaluate the adaptive autonomous behavior of a group of agents in a software simulation. Nevertheless, the long-term goal is to develop AA strategies that can be used both in real and artificial environments, and for heterogeneous agents.

8.2 Background and Related Work

This section gives an overview of the SAR domain and autonomy models, and it presents related work in multi-agent systems (MASs) coordination and cooperation.

8.2.1 The SAR Domain

The SAR domain has served as a testbed for multi-robot and multi-agent research in the past years. The RoboCup Rescue competitions have been established since the 2000s [5], where researchers would validate their findings either in simulation (Rescue Simulation League), or with real robotic platforms (Rescue Robot League) [6]. The Rescue Simulation League is divided further into the virtual and agent competitions. In regards to the virtual robot competition, attempts are being made to provide interfaces to ROS (Robot Op-

erating System) and Gazebo, due to the fact that the latter allow for better code-sharing among the community [7]. Kleiner et al. proposed the RMas-Bench [8], that could serve as a benchmark for coordination algorithms such as distributed constraint optimization problems (DCOP), whilst hiding low level details. Task allocation is usually expressed as the problem of assigning tasks to agents while also minimizing some cost function [9, 10]. The problem addressed in the present paper relates to how agents should interact with each other, i.e., when they should ask or give help based on their state, at any point in time. In particular, the ROS-based adaptive autonomous agent simulation developed in [4] was extended with the implementation of a SAR scenario.

Autonomy has been studied extensively in the literature, albeit there is no unified theory or general framework yet. Several autonomy changing schemes such as adaptive autonomy, adjustable autonomy, and mixed-initiative interaction, have been compared in the context of the coordination of large-scale teams of robots in a simulated Wilderness Search and Rescue (WiSAR) [11]. Agents with adaptive autonomy attempt at all times to keep the higher levels of autonomy. Furthermore, they do not go to the lowest level, in which the operator makes all the decisions. In the case of adjustable autonomy, the operator decides on the different levels of autonomy of agents. Whereas, mixed-initiative interaction allows both human and agent to make decisions on autonomy levels based on the circumstances. In these conditions, the third scheme yields the best results in terms of individuals found. Another approach uses a WiSAR scenario to evaluate the benefits of sliding autonomy in producing better paths (compared to two other methods that do not employ changes in autonomy), whilst not increasing the workload of the human operator [12].

In [13], a fire-fighter scenario is adopted in order to evaluate the reasoning mechanisms that allow fire-fighter agents to change their autonomy based on circumstances. Agent reasoning is based on two heuristic rules: (i) the more urgent a task becomes, the higher its priority, e.g., if the agent's health is running low then the task of taking a rest becomes more urgent as time passes, (ii) dedication to the organization, i.e., if the agent takes wrong information from the operator then it will take more initiative on its own.

8.2.2 Agent Cooperation and Coordination

Cooperation and coordination in MASs has been studied in other contexts that do not necessarily focus on the autonomy aspect. Theoretical approaches are concerned with providing taxonomies and definitions that set apart concepts such as cooperation, coordination, and collaboration [14] [15]. Recent years

have also seen a rise of interest in what are called open MAS (parts of the system come from competing third parties) [16] and pervasive systems (embedded sensors, actuators, etc.) modeled as MASs [17]. Surveys in the area have identified the following classes: stigmergy, chemical, physical, biochemical, field-based, and swarms [17, 18]. Attention has also been paid to design patterns that target the interaction among components in self-adaptive systems [19], among which the stigmergy pattern (interaction through the mediation of the environment), centralized pattern, and peer to peer pattern. Others propose a framework in which specifications related to permissions and obligations among agents are adjusted at run-time [16]. Earlier works use a central coordinator, e.g., Kaa [20], which approves or rejects changes in permissions suggested by agents, and refers to an operator in case a decision is not reached. It is assumed that when such permissions change, so does the autonomy of agents.

8.3 The Adaptation Strategy

In this Section, the agent model proposed in [3], is revised, and the proposed adaptation strategy for AA is described.

8.3.1 The Agent Model

An agent is defined as a computer system that perceives and acts in its environment through its sensors and actuators respectively, and is capable of autonomous action [21]. Physical agents in addition to the previous characteristics, are able to manipulate the physical world through their effectors (e.g., legs, wheels, or manipulators). Any such agent needs to possess the necessary skills, abilities, knowledge, and resources to perform a particular task [22]. Moreover, an agent's operation is limited by its available resources, such as the battery level (e.g., for a robot), the allowed power consumption, or the available computational resources (e.g., for a software agent). Therefore, the agent can be described in terms of its (i) battery level, (ii) equipment (sensors, motors, manipulators, and actuators), (iii) skill/ability set, and (iv) knowledge. The agent's knowledge can refer to what the agent knows about itself, other agents, and the environment.

The agent model adopted in this paper is a finite-state machine, shown in Figure 8.1. The agent has five states: *idle*, *execute*, *interact*, *out_of_order* and *regenerate*. All agents start their operation in the *idle* state. In this state, a

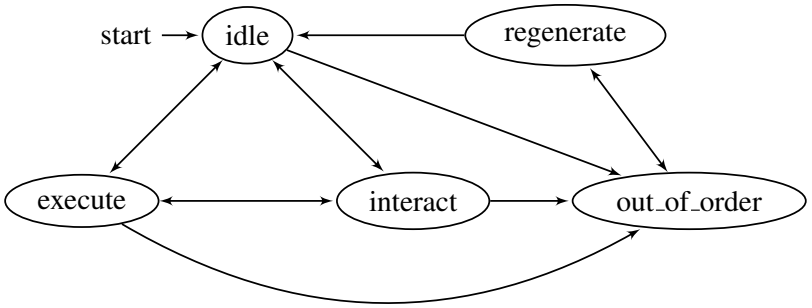


Figure 8.1: The proposed agent model composed of five states [3].

task can be generated. Consequently, the agent will switch to *execute*, where either it executes its task, or it can decide to interact with other agents asking for assistance. When the task is completed, the agent returns to *idle*. When an agent receives a request from another agent, it will switch to *interact*, where it will decide to accept or discard it. In the former case, the agent will put its current task, if any, back into a FIFO queue, and start the new task. Otherwise it will return to its previous state, either *idle* or *execute*. It is assumed that these queues are infinite long. Moreover, any agent cannot execute more than one task at the same time. When the energy level of an agent is low, it will switch to *out_of_order*, and soon after to *regenerate*, during which the recharging process takes place. If *regenerate* is successful, the agent goes to *idle*, and continues its normal operation. If *regenerate* fails, it will go to *out_of_order*. In the current setting, the agent is always assumed to regenerate successfully.

8.3.2 Willingness to Interact

Adaptive autonomous behavior is modeled through the willingness to interact, composed of two components referred to as the willingness to ask for, and give help [3]. The willingness to ask for help represents the likelihood with which an agent will ask another agent for help during the execution of a task (*execute* state). Whereas, the willingness to give help represents the likelihood with which an agent will provide help upon a request from another agent (*interact* state). In this paper, the goal of the agent is to decrease the disposition of asking for help and increase the disposition of giving help, when possible, promoting the cooperation among agents, assuming that cooperation can improve

the performance of the MAS [23].

An agent computes the *willingness to ask for help* at time t, $\gamma_t \in [0,1]$, by applying a correction to its initial value, as

$$\gamma_t = \text{sat}(\gamma_0 + u_t), \tag{8.1}$$

where, $\text{sat}(x) := \min(\max(x,0),1)$, and u_t is the *adaptive correction* at time t, computed as

$$u_t = \sum_{i=1}^{n} \phi_i f_t^{i\gamma}, \tag{8.2}$$

where n is the number of factors, ϕ_i, $i = 1,\ldots,n$, are weights (constant or calculated at runtime), such that $\phi_i \in [0,1]$, $\sum_{i=1}^{n} \phi_i = 1$, and $f_t^{i\gamma}$, are the considered factors at time t.

An agent computes the *willingness to give help* at time t, $\delta_t \in [0,1]$, by applying a correction to its initial value, as

$$\delta_t = \text{sat}(\delta_0 + v_t). \tag{8.3}$$

The correction v_t integrates the effect of several factors, and it is computed as

$$v_t = \sum_{i=1}^{n} \zeta_i f_t^{i\delta} \tag{8.4}$$

where n is the number of factors, ζ_i, $i = 1,\ldots,n$, are weights (constant or calculated during runtime), such that $\zeta_i \in [0,1]$, $\sum_{i=1}^{n} \zeta_i = 1$, and $f_t^{i\delta}$ are the considered factors at time t. In the following paragraphs, the time subscript is omitted for the sake of simplicity.

The k-th factor f^{k*} is updated at every iteration based on the current measurements ψ_k of relevant quantities, e.g., the battery level of the agent, the accuracy of the agent in carrying out a specific task, etc, and on a minimal acceptable value $\psi_{k\min}$. The update rules for the willingness to give and ask for help are the following:

$$f^{k\gamma} = \begin{cases} \psi_{k\min} - \psi_k, & \psi_k > \psi_{k\min} \\ (1-\alpha)(\psi_{k\min} - \psi_k) + \alpha, & \psi_k \leq \psi_{k\min} \end{cases} \tag{8.5}$$

$$f^{k\delta} = \begin{cases} \beta(\psi_k - \psi_{k\min}) & \psi_k > \psi_{k\min} \\ \beta(1-\alpha)(\psi_k - \psi_{k\min}) - \alpha, & \psi_k \leq \psi_{k\min} \end{cases} \tag{8.6}$$

where $\alpha \in \{0,1\}$ and $\beta \in \{-1,1\}$ are *control parameters*, and influence the calculation of different factors based on the desired behavior. In general $f^{k*} \in$

$[-1,1]$. When $f^{k\gamma} \to 1$, the likelihood of asking for help is high, while when $f^{k\gamma} \to -1$ the likelihood of asking for help is low. Analogously, when $f^{k\delta} \to 1$, the likelihood of giving help is high, while when $f^{k\delta} \to -1$ the likelihood of giving help is low. Note that, f^{k*} does not necessarily dominate all the other factors. As a result, a_i might still decide to ask for or give help when the threshold is exceeded.

The main rationale of (8.5) and (8.6) is that the factors should be adjusted proportionally to the distance between the current measurement ψ_k and the threshold value $\psi_{k\min}$. In particular, when $\psi_k \geq \psi_{k\min}$, agent a_i does not need help to complete its task, so it will decrease the likelihood of asking for help, while, at the same time, it will increase its willingness to give help to other agents.

In order for any factor to dominate all the factors (f^{k*}) given certain conditions, the corresponding weights ϕ_i in (8.2), or the corresponding weights ζ_i in (8.4), should be set to 1. Since $\sum_{i=1}^{n} \phi_i = 1$ and $\sum_{i=1}^{n} \zeta_i = 1$, the weights for all other factors are set to zero. If two or more of the factors f^{k*} is equal to one, then the weight of one of them is set to one, whilst all others are set to zero. When no factor is equal to one, weights for all factors are all equal and calculated as $\phi_i = 1/n$ and $\zeta_i = 1/n$.

8.3.3 Factor description

In order to calculate the corrections u and v, nine factors ($n = 9$) have been identified in [3] as relevant in the process of deciding when to ask and give help during runtime. All the factors follow the update rules (8.5) and (8.6).

The **battery factor** f^{1*} is the amount by which agent a_i's battery level $\psi_1 = b \in [0,1]$ will affect its willingness to interact. f^{1*} is computed with $\psi_{1\min} = b_{\min} + b_{\tau_j}$, where $b_{\min} \in [0,1]$ is the minimum battery threshold for which a_i can operate in the *execute* state, $b_{\tau_j} \in [0,1]$ is the amount of energy required by the task τ_j, $\alpha = 1$, and $\beta = 1$.

The **knowledge factor** f^{2*} is the amount by which the agent's confidence on its knowledge level $\psi_2 \in [0,1]$ will affect its willingness to interact. f^{2*} is computed with $\psi_{2\min} = 0$, $\alpha = 1$, and $\beta = 1$.

The **skill/ability factor** f^{3*} is the amount by which the agent's skill efficiency level $\psi_3 \in [0,1]$ in performing a task will affect its willingness to interact. f^{3*} is computed with $\psi_{3\min} = 0$, and $\alpha = 1$.

The **equipment factor** f^{4*} is the amount by which the agent's equipment accuracy level $\psi_4 \in [0,1]$ for performing a task will affect its willingness to interact. f^{4*} is computed with $\psi_{4\min} = 0$, $\alpha = 1$, and $\beta = 1$.

The **resource factor** f^{5*} is the amount by which the agent's resource quality level $\psi_5 \in [0,1]$, i.e., how much the type of tools in the agent's possession fit the task to be performed, will affect its willingness to interact. f^{5*} is computed with $\psi_{5\min} = 0$, $\alpha = 1$, and $\beta = 1$.

The **performance factor** f^{6*} is the amount by which the agent's current performance level $\psi_6 \in [0,1]$ will affect its willingness to interact. Performance is a general indicator of how well an agent's outcome is with respect to the tasks attempted in the past. f^{6*} is computed with $\psi_{6\min} \in [0,1)$, $\alpha = 0$, and $\beta = 1$.

$f^{7\gamma}$ is the **task progress factor**, covers the progress towards completion $\psi'_7 \in [0,1]$ of the current task, while $f^{7\delta}$ is the **task trade-off** factor, and it covers the trade-off $\psi_{7''} \in [0,1]$ between a task τ_j currently in execution, and a new task proposed by another agent τ'_j. The two factors are computed as (8.5) and (8.6), with $\psi_{7\min} \in [0,1)$, $\alpha = 0$, and $\beta = 1$.

The **environment factor** f^{8*} is the amount by which the agent's likelihood of succeeding in its environment $\psi_8 \in [0,1]$ will affect its willingness to interact. This likelihood could be estimated based on the difficulties an agent perceives in its environment, e.g., obstacles, harsh conditions and so on. f^{8*} is computed with $\psi_{8\min} \in [0,1)$, the acceptable level for this likelihood, $\alpha = 0$, and $\beta = 1$.

The **collaboration factor** f^{9*} is the amount by which the agent's estimated likelihood of a successful collaboration with another $\psi_9 \in [0,1]$ will affect its willingness to interact. f^{9*} is computed with $\psi_{9\min} \in [0,1)$, the acceptable level of such likelihood, $\alpha = 0$, and $\beta = -1$, through equation (8.6). It can be observed that this factor affects the willingness to give and ask for help in the same way.

In this paper, only factors $f^{1-5\gamma}$ are considered to be dominant, because it is assumed that in case either battery, abilities, equipment, knowledge, or tools are not adequate then the agent should not attempt the task autonomously. Moreover, it is assumed that the agent is able to estimate the different factors.

8.4 Simulation setup

The following paragraphs describe the implementation of the SAR scenario, and the implementation of the agent model. ROS [24] has been used as the underlying communication middleware. The agents and environment are implemented as ROS nodes (executables), and communicate with each other through the ROS middleware.

8.4.1 Scenario instantiation

The environment is a 2D space with a specific width $grid_w = 30$ and height $grid_h = 30$, and is generated at the beginning of a simulation. Agents can move in this space, from a point A to a point B, according to the shortest Euclidean path between two points. Obstacles and collisions are not considered because they are not crucial to the level of interactions discussed here. In this space, a fixed number of fires ($n_f = 40$) is generated at random locations (x, y), with the same intensities $I \in \{75, 100\}$ (the value of the intensities depends on the difficulty of the simulation). The intensities are kept fixed in time for simplicity. After a fire is extinguished, the trapped individuals (who lie at the same location as fires) become visible and can thereafter be saved. The number of these individuals at a particular location is the same for each location, $n_v \in \{15, 20\}$ (depending on the difficulty of the simulation). Moreover, two base stations, one for fire brigades and one for ambulances, are initiated at two random locations in the grid.

The environment node publishes continuously the current state of fire intensities, fire status (active/inactive), number of trapped individuals, and location in the grid. Fire extinguishing, and individual extraction is simulated by having the appropriate agent make a ROS service call to the environment node. As a result, the corresponding variable (fire intensity or number of trapped individuals) will decrease by a predefined $step = 1$.

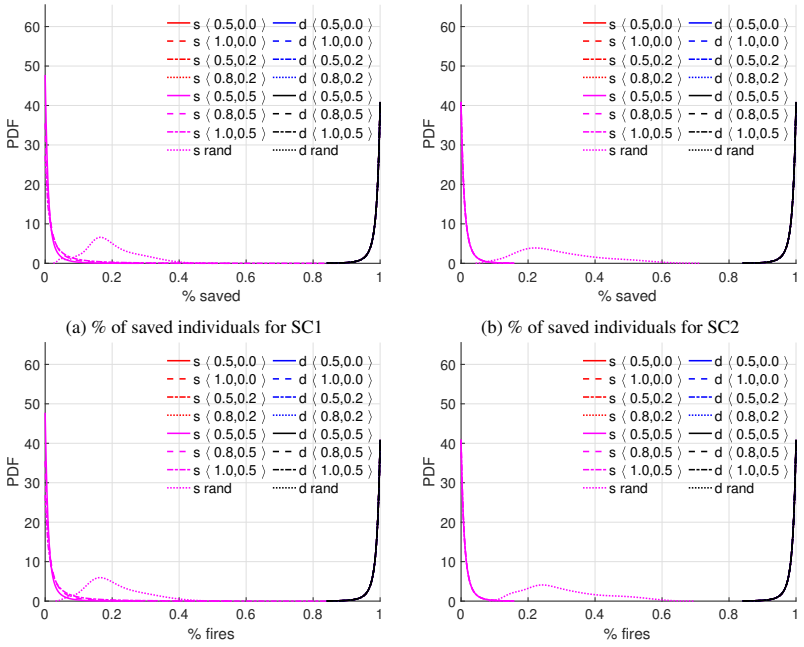
8.4.2 Agent instantiation

Agents are implemented as ROS nodes. Their interactions with each other are realized through the ROS publish/subscribe mechanism (for broadcasting), and action server mechanism (for one to one calls). There are three types of agents: fire brigades, ambulances, and police. Fire brigade agents put out fire. Ambulances extract and carry individuals to safety. Police agents detect the fires/victims within their range, and broadcast this message to the others. They all broadcast their identity and abilities (e.g., fire extinguishing and transportation), as a result they are known to one another.

Throughout the whole the simulation, agents perform continuous Levy walks in the generated 2D space, when in the *idle* state. The Levy walk algorithm was implemented as it has been considered an efficient strategy for search algorithms independent of the target distribution [25]. During these walks, agents publish their location to the environment node, and are able to detect fires/individuals if they are within a specified Euclidean distance

($d_v = 10\% \times grid_w = 10\% \times grid_h = 3$). Trapped individuals are not visible as long as the status of a fire is active. Fire brigade agents can also be notified by police agents about the existence of a fire, as a result their information is not restricted to what is in their range of visibility at a particular time. If there are visible fires, a fire brigade agent will generate a corresponding task for extinguishing it. The task will have as many iterations as the fire intensity. If more than one fire location is visible, then one is chosen randomly and pursued. Similarly, if there are visible trapped individuals, ambulance agents will pick one randomly and generate a corresponding task with the number of iterations being equal to the number of individuals. When a task is generated, agents will walk to the location of interest. They are assumed not to need help for reaching any location in the 2D space. No initial plan, or explicit global coordination is assumed. Nevertheless, it might happen that several agents go for the same fire, or individuals to extract. When an agent decides to ask for help, it will ask the appropriate agents it knows one by one, until either a task succeeds, or the list of agents to ask is exhausted. Furthermore, an agent that is already waiting upon another for a task, will discard on the spot any help requests it receives. This is to ensure that agents do not wait on one another pointlessly.

Each agent starts at the same level of battery, at $\psi_1 = 1$, corresponding to the maximum available energy of each agent. During task execution, the agent's energy level is decreased with a certain level, in each iteration step. Also, when an agent moves to the location of interest, its energy level is reduced proportionally to the covered distance. There is a low threshold $b_{\min} = 0.3$ under which the agent will go to *out of order* and thereafter recharge. Agents are assumed to have the necessary knowledge for performing tasks, with knowledge $\psi_2 = 1$. The same assumptions hold for abilities ($\psi_3 = 1$), and equipment ($\psi_4 = 1$). Initially, agents have the necessary resources depending on their function, i.e., fire brigades have water $r_w = 25$ units of water, whereas ambulances have space for $r_s = 5$ individuals inside their vehicle. When the agent has enough resources to extinguish the fire or to carry an individual, the respective resource factor will be $\psi_5 = 1$, and it will be $\psi_5 = 0$ otherwise. During the run of each task, these resources decrease according to their usage. After that a task is completed, each agent will move to its corresponding base station and reset its own resources. Agents are assumed to be able of estimating: environmental risk, potential collaborator risk, their own performance, progress of their current task, and task trade-off between two tasks. In these simulations, the likelihood of success in the environment is kept constant for simplicity. The likelihood of success with a potential collaborator is estimated



(a) % of saved individuals for SC1

(b) % of saved individuals for SC2

(c) % of extinguished fires for SC1

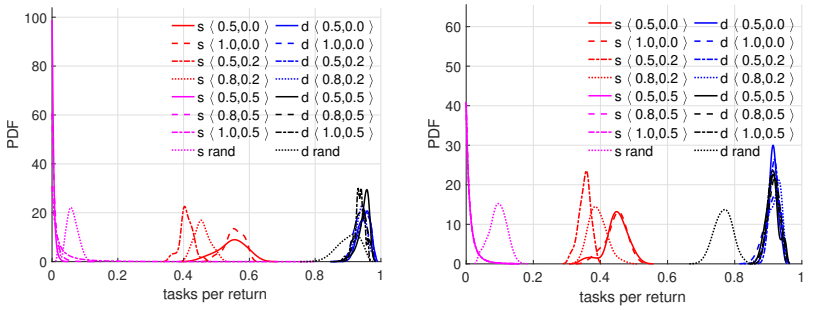
(d) % of extinguished fires for SC2

Figure 8.2: PDFs of the results for the *% saved* and *% fires* metrics, in SC1–2.

based on the the percentage of past successful interactions. Agents calculate their performance through $\psi_6 = {}^{t_c}/_{t_{\text{tot}}}$, where t_c is the number of completed tasks, and t_{tot} is the total number of attempted tasks. Task progress is calculated by ${}^{it_c}/_{it_{\text{tot}}}$, where it_c is number of iterations completed, and it_{tot} is the total number of iterations. Finally, the task trade-off for two tasks τ_1 and τ_2 is calculated through ${}^{R_2-R_1}/_{R_2+R_1}$, where R_1 and R_2 are the respective rewards. Finally, the thresholds for these factors are set to 0 for simplicity, thus $\psi_{2-8\,\text{min}} = 0$.

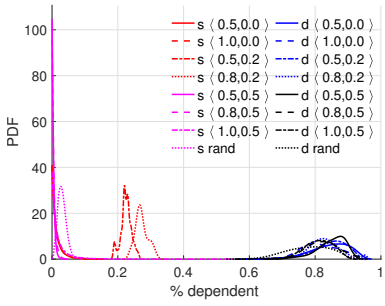
8.5 Results

The hypothesis, evaluated through simulations, is formulated as follows: "Agents with adaptive autonomy perform better than agents that do not display such behavior, in the context of a simulated search and rescue scenario". Performance is assessed across several metrics (Section 8.5.1). There are two

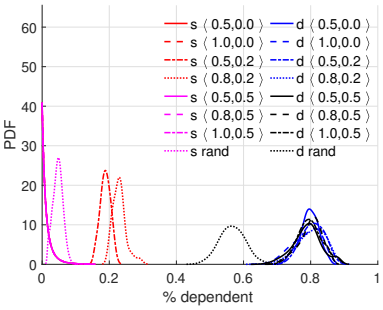


(a) Task per return to base for SC1

(b) Task per return to base for SC2



(c) % dependent for SC1



(d) % dependent for SC2

Figure 8.3: PDFs of the results for the *task per return* and *% dependent* metrics, in SC1–2.

modes of execution for any of the simulation runs, (i) static mode, in which the components of the willingness to interact are set in the beginning, do not change during runtime, and are independent of factors, and (ii) dynamic mode, in which the components of the willingness to interact change during runtime (every update is done against the same initial value specified for each component) based on factors. For both modes, the same couples $\langle \delta_0, \gamma_0 \rangle$ for the components of the willingness to interact is used. In the evaluation, the following couples are considered: $\langle 0.5, 0.0 \rangle$, $\langle 1.0, 0.0 \rangle$, $\langle 0.5, 0.2 \rangle$, $\langle 0.8, 0.2 \rangle$, $\langle 0.5, 0.5 \rangle$, $\langle 0.8, 0.5 \rangle$, $\langle 1.0, 0.5 \rangle$, and a random configuration. In the first seven configurations, each agent in the population is initialized with the same willingness to interact. Whereas, in the random configuration, each agent is initialized with different random values for δ_0 and γ_0. Furthermore, simulations, for each configuration and both modes, are run across two scenarios which differ in dif-

ficulty (the random values for the willingness to interact are the same within a scenario, but differ across scenarios). Difficulty is defined as the ratio between the required resources on each site ($w \in \{75,100\}$ and $s \in \{15,20\}$) and the agent's initial resources ($r_w = 25$ or $r_s = 5$). All simulations were ran on a system with 24 cores, 8GB RAM, with Ubuntu 14.04 LTS operating system, and ROS Indigo[1].

8.5.1 Evaluation metrics

The quality of the obtained results is evaluated according to four metrics, as follows: (i) percentage of individuals saved in the simulation with respect to the total number of trapped individuals (*% saved*), (ii) percentage of fires extinguished in the simulation with respect to the total number of fires distributed in the 2D space (*% fires*), (iii) number of completed tasks per return to the base (*task per return*), (iv) percentage of dependent completed tasks with respect to all dependent tasks attempted (*% dependent*). A task is dependent when an agent needs assistance for its completion. The results are averaged over the whole population of agents and over the number of repetitions (30), for each simulation instance, i.e., for any combination of scenario difficulty and initial configuration of the willingness to interact.

8.5.2 Numerical results

Simulation results are displayed in Figures 8.2 and 8.3 as probability density functions (PDFs) obtained with 30 different runs of the considered scenarios. Each figure contains the outcomes with respect to one of the metrics, across the two difficulty scenarios (SC). Each sub-figure contains the outcomes for the eight initial configurations and the two modes of execution (the static and dynamic strategies). The red and magenta lines refer to the static case (indicated with 's' in the legend), while the blue and black lines refer to the dynamic strategy (indicated with 'd' in the legend).

Regarding the percentage of saved individuals, in both difficulty scenarios, all dynamic agents and static agents with $\gamma_0 \in \{0.0, 0.2\}$ manage to save all individuals within the allotted time (Figures 8.2a-8.2b). Whereas, for static agents with $\gamma_0 = 0.5$ is nearly 0, and for the random configuration on average 20%. Similar considerations apply for the percentage of extinguished fires

[1]The source code used in producing the results displayed in this paper is publicly available at https://github.com/gitting-around/gitagent-sar.git.

(Figures 8.2c-8.2d). This is to be expected, because if by the time the simulation ends, only half of the fires have been extinguished then only half of the individuals could have been saved.

Dynamic agents achieve higher percentage of completed tasks with respect to returns to the base, (Figures 8.3a-8.3b). In SC1–2, all the static agents perform worse than the dynamic counterparts. Nevertheless, among static agents, the ones with $\gamma_0 = 0.0$ achieve better results than others. Whereas in the dynamic mode, the random configuration seems to produce slightly worse results than the other configurations. The results are consistent across the two scenarios.

Dynamic agents perform noticeably better as compared to their static counterparts with respect to the percentage of completed dependent tasks (Figures 8.3c-8.3d). The outcomes are stable among different configurations and scenarios, with the dynamic random configuration always performing slightly worse than others (especially visible in SC2). In the static mode, better results are obtained for configurations where $\gamma_0 = 0.2$. These results agree with previous work [4], which shows that when agents need to depend on each other all the time, and are highly likely to give help, the performance of the system will degrade.

8.6 Conclusion

The obtained results indicate that adaptive autonomous behavior can positively impact the performance of a population agents, in a context of a SAR scenario. Performance is assessed with respect to four metrics: percentage of individuals saved, percentage of fires extinguished, percentage of completed tasks with respect to the returns to base, percentage of completed dependent tasks. Among the two difficulty scenarios, the adaptive autonomous agents maintain a stable behavior, seemingly independent of the initial configuration. Whereas, static agents are quite dependent on the initial configuration. The results displayed in the paper can have a dependency on the platform used for the simulations. In more powerful platforms, all agents can perform better overall, and in less powerful ones, they are expected to have poorer performance overall, i.e., less individuals saved and extinguished fires within the allotted time for the simulation.

There are several directions for future work. Firstly, an agent reasons on each iteration whether to ask for help. This is particularly penalizing in the static case, where even with low willingness to ask for help on each iteration,

most tasks end up being dependent tasks. Thus, the possibility that the agent decides on whether to ask for help once in a couple of iterations, needs to be investigated. Secondly, an agent can accept to help another agent, even if it does not fulfill energy, abilities, equipment, knowledge, or resource requirements. Consequently, it will ask for help a third agent. It is needed to compare this design choice, with the other option that involves the agent declining upfront to help in such scenario. Finally, cues regarding the grade of dependencies within the population can be used as an additional factor, allowing the agent to regulate its own behavior from the global perspective as well.

Bibliography

[1] K. Suzanne Barber, Anuj Goel, and Cheryl E. Martin. Dynamic adaptive autonomy in multi-agent systems. *Journal of Experimental & Theoretical Artificial Intelligence*, 12(2):129–147, 2000.

[2] Cristiano Castelfranchi. Founding agent's 'autonomy' on dependence theory. In *ECAI*, pages 353–357, 2000.

[3] Mirgita Frasheri, Baran Çürüklü, and Mikael Ekström. Analysis of perceived helpfulness in adaptive autonomous agent populations. *LNCS Trans. on Computational Collective Intelligence*, 2017.

[4] Mirgita Frasheri, Baran Çürüklü, and Mikael Ekström. Comparison between static and dynamic willingness to interact in adaptive autonomous agents. In *ICAART*, 2018.

[5] Satoshi Tadokoro et al. The robocup-rescue project: A robotic approach to the disaster mitigation problem. In *ICRA*, volume 4, pages 4089–4094, 2000.

[6] Raymond Sheh, Sören Schwertfeger, and Arnoud Visser. 16 years of robocup rescue. *KI-Künstliche Intelligenz*, 30(3-4):267–277, 2016.

[7] Arnoud Visser, Francesco Amigoni, and Masaru Shimizu. The future of robot rescue simulation workshop. *Benelux AI Newsletter*, 30(2), 2016.

[8] Alexander Kleiner et al. Rmasbench: benchmarking dynamic multi-agent coordination in urban search and rescue. In *AAMAS*, pages 1195–1196, 2013.

[9] James Parker et al. Exploiting spatial locality and heterogeneity of agents for search and rescue teamwork. *Journal of Field Robotics*, 33(7):877–900, 2016.

[10] D. Di Paola et al. Optimal control of time instants for task replanning in robotic networks. In *ACC*, pages 1993–1998, 2016.

[11] Benjamin Hardin and Michael A Goodrich. On using mixed-initiative control: A perspective for managing large-scale robotic teams. In *HRI*, pages 165–172, 2009.

[12] Lanny Lin and Michael A Goodrich. Sliding autonomy for UAV path-planning: Adding new dimensions to autonomy management. In *AAMAS*, pages 1615–1624, 2015.

[13] Bob van der Vecht, Frank Dignum, and JJ Ch Meyer. Autonomy and coordination: Controlling external influences on decision making. In *WI-IAT*, volume 2, pages 92–95, 2009.

[14] Fatemeh Golpayegani et al. Multi-agent collaboration for conflict management in residential demand response. *Comp. Comm.*, 96:63–72, 2016.

[15] H van Dyke Parunak et al. A design taxonomy of multi-agent interactions. In *AOSE*, pages 123–137, 2003.

[16] Alexander Artikis et al. Specifying and executing open multi-agent systems. In *Social Coordination Frameworks for Social Technical Systems*, pages 197–212. Springer, 2016.

[17] Franco Zambonelli et al. Developing pervasive multi-agent systems with nature-inspired coordination. *Pervasive and Mobile Computing*, 17:236–252, 2015.

[18] Stefano Mariani and Omicini Andrea. State-of-the-art and trends in nature-inspired coordination models. *The IEEE Intelligent Informatics Bullettin*, 18(2):35–40, 2017.

[19] Mariachiara Puviani, Giacomo Cabri, and Franco Zambonelli. Patterns for self-adaptive systems: agent-based simulations. *EAI Endorsed Trans. on Self-Adaptive Systems*, 1:1–15, 2015.

[20] Jeffrey M Bradshaw et al. Kaa: policy-based explorations of a richer model for adjustable autonomy. In *AAMAS*, pages 214–221, 2005.

[21] Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152, 1995.

[22] Matthew Johnson et al. Coactive design: Designing support for interdependence in joint activity. *Journal of Human-Robot Interaction, 3 (1)*, 2014.

[23] Sarit Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94(1):79 – 97, 1997. Economic Principles of Multi-Agent Systems.

[24] Morgan Quigley et al. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[25] ME Wosniack et al. Robustness of optimal random searches in fragmented environments. *Physical Review E*, 91(5), 2015.

Chapter 9

GLocal: A Hybrid Approach to the Multi-Agent Mission Re-Planning Problem

Authors: Mirgita Frasheri, Branko Miloradović, Baran Çürüklü, Mikael Ekström, and Alessandro V. Papadopoulos.
Venue: Technical Report[1]

[1]Submitted at the 1st International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS'20)

Abstract

Multi-robot systems can be prone to failures during plan execution, depending on the harshness of the environment they are deployed in. As a consequence, initially devised plans may no longer be feasible, and a re-planning process needs to take place to re-allocate any pending tasks. Two main approaches emerge as possible solutions, a global re-planning technique using a centralized planner that will redo the task allocation with the updated world state information, or a decentralized approach that will focus on the local plan reparation, i.e., the re-allocation of those tasks initially assigned to the failed robots. The former approach produces an overall better solution, while the latter is less computationally expensive. The goal of this paper is to exploit the benefits of both approaches, while minimizing their drawbacks. To this end, we propose a hybrid approach that combines a centralized planner with decentralized multi-agent planning. In case of an agent failure, the local plan reparation algorithm tries to repair the plan through agent negotiation. If it fails to re-allocate all of the pending tasks, the global re-planning algorithm is invoked, which re-allocates all unfinished tasks from all agents. The hybrid approach was compared to planner approach, and it was shown that it improves on the makespan of a mission in presence of different numbers of failures, as a consequence of the local plan reparation algorithm.

9.1 Introduction

Automated planning is the process of defining a set of actions for an autonomous system to achieve a prescribed set of goals. The problem of automated planning has been a central research topic in artificial intelligence for the last decades [1], and it has become particularly relevant in the context of Multi-Agent Systems (MASs) [2].

Several approaches have been proposed to address the automated planning problem, and they are divided into two major categories: centralized and distributed planning. While there are clear advantages of centralized algorithms related to optimality of the computed plan with respect to an objective function, the curse of dimensionality has been the main limitation of such approaches. On the other hand, distributed algorithms provide a more robust alternative towards faults [3, 4], safety [5, 6], and security [7, 8].

The trade-off between the optimality of centralized solutions, and the flexibility and robustness to potential failures of distributed approaches is the main focus of this paper. In addition, this trade-off is studied in the context of agent or robot failures [2]. When a failure occurs, the centralized approach will compute a new plan based on the current conditions, referred to as re-planning, whereas the decentralized approach will rely on self-organization, which allows agents to perform a local plan reparation.

In this paper, a novel hybrid approach for multi-agent automated planning, GLocal, is proposed. GLocal exploits the advantages of both approaches, i.e., optimality and robustness, while limiting their inherent disadvantages. In particular, this paper investigates what is the effect of failures in MAS automated planning, and it shows the robustness of the proposed hybrid approach. When a failure occurs, agents in the MAS attempt to repair the plan locally, by negotiating with one another over the assignment of the pending tasks, i.e., tasks initially assigned to the failed agent. Agent collaboration is shaped by their willingness to interact, which captures the utility of being assigned to a given task. In case at least one task remains un-allocated, agents make a request to the centralized global planner for a re-plan, and as a result they switch from a local to a global strategy. More specifically, this paper focuses on the following research questions (RQs)

RQ1 How does the number of replans from a centralized planner impact on the overhead and quality of the solution of a MAS?

[2] In this paper, the terms agent and robot will be used interchangeably.

RQ2 How can the agents minimize the number of calls to the planner by collaborating with one another?

These questions are addressed through computer simulations, where the GLocal approach was compared to a planner-only approach, for different number of failures, as well as sizes of problem instances. The results reveal that GLocal produces solutions with shorter mission make-spans in the presence of failures, as a consequence of reduced calls to the centralized global planner.

The rest of the paper is organized as follows. Section 9.2 describes the background of the paper, and the problem addressed in this paper is presented in Section 9.3. Section 9.4 describes the design of the MAS that controls the behavior of the robots, while Section 9.5 presents the centralized global planner for the agents. Section 9.6 describes the simulation setup, and Section 9.7 presents the experimental results. Finally, Section 9.8 discusses the related work, and Section 9.9 concludes the paper.

9.2 Background

In real world applications, the operation of agents or robots can be disrupted by environments changes, which occur regardless of the agent's activities. Disruption can also occur due to unforeseen events such as faulty sensors or actuators, thus making an agent incapable of performing certain tasks. Additionally, the goals, toward which such agents are working for, can themselves be subject to change. In order to cope in such complex situations, distributed and continual planning approaches have been proposed, that (i) distribute the planning process among a group of agents, and (ii) allow for planning to be an incremental process that happens continuously during the operation of agents, as well as (iii) combined approaches for distributed continual planning (DCP) [9]. Multi-agent planning has been defined as the problem of creating a plan for and by a group of agents [10]. Furthermore, five stages of MAP have been identified such as goal allocation to agents, refinement of goals into sub-tasks, sub-task scheduling by considering other constraints, communication of planning decisions, and plan execution.

Depending on the perspective, distributed planning can refer to either cooperative distributed planning (CDP), also known as cooperative and distributed multi-agent planning (MAP)[3], or to negotiated distributed planning [11]. In

[3] A recent survey on cooperative and distributed MAP, referred to also as multi-agent coordination of actions in decentralized systems, provides a taxonomy of existing approaches in the

the former view, the goal is to create a global plan, whereas in the latter the emphasis is on the agents ability to fulfil their own local objectives. While a CDP focuses on issues as plan representation and generation, task allocation, communication and coordination, in the scope of an NDP, the focus is on collaboration and cooperation between agents. Continual planning on the other hand allows agents to revise their plans during operation as unforeseen events occur. Examples include reactive planning systems, where an agent considers only the next step and does not look ahead further in the future; and flexible plan execution systems, which allow for some look ahead, and delay sketching out the detailed plan as much as possible.

Centralized planning implies that decisions are not made independently and locally, but rather holistically at a global level. Utilizing centralized planning to solve multi-agent planning problems is a widely accepted approach. Landa-Torres *et al.* [12] used a centralized planner, based on the evolutionary algorithms, to solve an underwater multi-agent mission planning problem for a swarm of autonomous underwater vehicles. Similarly, the solution to the problem of mission planning for a swarm of unmanned aerial vehicles was presented by Ramirez-Atencia *et al.* [13]. The problem is modeled as a constraint satisfaction problem and solved using a multi-objective Genetic Algorithm (GA). A different approach to a similar problem is taken by Karaman Sertac *et al.* [14] where process algebra is used to model the problem that is later solved with the GA.

This paper is concerned with the investigation of methods that combine centralized and negotiated distributed planning approaches in order to optimize the execution of plans in a failure prone context, simultaneously increasing the robustness of the system by allowing agents to perform a local plan reparation online.

9.3 Problem Formulation

The problem that is being addressed in this paper is a relaxed version of the Extended Colored Traveling Salesperson Problem (ECTSP) [15]. The original problem is simplified by the removal of the precedence constraints among tasks.

Assume a set of n tasks, $v \in \mathcal{V} := \{v_1, v_2, \ldots, v_n\}$, m agents, $s \in \mathcal{S} := \{s_1, s_2, \ldots, s_m\}$, and k capabilities, $c \in \mathcal{C} := \{c_1, c_2, \ldots, c_k\}$ where $m, n, k \in \mathbb{N}$.

literature based how they deal with issues such as agent distribution, computational process, plan synthesis schemes, communication mechanisms, heuristic search, and privacy preservation [2].

Each agent $s \in \mathcal{S}$ has a set of capabilities $\mathcal{C}_s \subseteq \mathcal{C}$ assigned to it. Each task $v \in \mathcal{V}$ requires one capability in order to be successfully completed. A capability matrix of an agent s, $\mathcal{A}_s \in \{0,1\}^{n \times n}$, can be defined as:

$$a_{ijs} = \begin{cases} 1, & f_c(v_i) \in \mathcal{C}_s \wedge f_c(v_j) \in \mathcal{C}_s \\ 0, & \text{otherwise,} \end{cases} \tag{9.1}$$

Agents are allowed to move in a 2D space Z, that is represented as a continuous map, and are able to communicate with one another with broadcast, without limitation in range.

The problem consists of allocating n tasks to m agents with respect to given constraints in the form of agent capabilities and task requirements for such capabilities in order to minimize the make-span of a mission.

Objective function The goal is to complete all the tasks in the environment while minimizing mission's duration, even in presence of one or more agent failures. Agent failures may be due to a physical failure of the robot performing a specific task, or of the equipment that is required to perform said task.

In MASs, a mission can involve optimization of many different parameters. Commonly, mission duration is minimized, however, a duration of a mission can be defined in various ways [16]. The objective function used in this work tends to minimize the duration between the starting time of the first task and end time of the last task over all agents in the mission. This objective function is also known as "minMax", as it minimizes the maximum duration of an agent's makespan over all agents.

9.4 Agent Design

The agent design consists of three core aspects, namely (i) the architecture comprising of a finite state machine which captures the different behaviours of an agent, (ii) the willingness to interact abstraction and its role in shaping collaborative behaviour, and (iii) the interaction protocols used in any collaboration.

9.4.1 Agent Architecture

Each agent is designed as a finite state machine composed of four states, *idle*, *interact*, *execute*, *interact & execute* (Fig. 9.1), and starts its operation in *idle*.

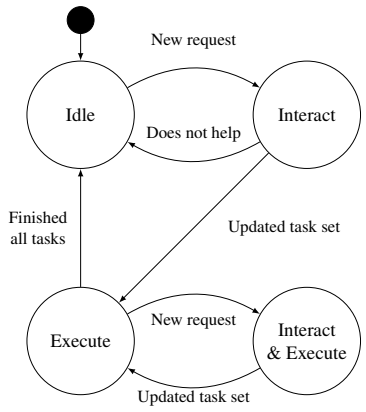


Figure 9.1: Agent operation state machine.

Agents in *idle* are not committed to any task. Note that, depending on the application, other types of behaviours could be implemented in this state. Agents are able to create tasks on their own, e.g., reaching an object location after detecting its presence. Additionally, they are able to get a list of tasks by another entity, e.g., a centralized planner; this is the case studied in this paper. When an agent commits to performing a task, it switches to *execute* and proceeds with the task execution. Once an agent completes all its tasks, it reverts back to the *idle* state. As soon as a new task is detected or received, an agent, residing either in *idle* or *execute*, switches to *interact*, or *interact & execute*, respectively. Thereafter, a negotiation process is initiated with other agents in order to allocate every new task. The outcome of the negotiation is zero or more agents assigned to perform each task. An agent with no assignment at the end of the negotiation goes back to either *idle* or *execute*, respectively, i.e., the state it was in before the negotiation round.

9.4.2 Willingness to Interact

The collaborative behaviour of an agent a_i is determined by its willingness to interact $w_i(t) \in [-1,1]$, i.e., the likelihood of asking and giving help to other agents at time t. A positive willingness indicates that a_i is able to help others $w_i(t) > 0$, whilst a negative willingness indicates that a_i needs help performing

its tasks $w_i(t) < 0$, with $w_i(t) = -1$ indicating that an agent must ask for help at time t, and $w_i(t) = 0$ denotes a neutral disposition. The willingness is affected by both the state of an agent, which captures the general attitude towards potential collaboration with others (explored in previous work [17]), and by the properties of the specific task considered during any negotiation, namely the utility of performing such task.

In this paper the focus is solely on how the utility of performing a particular task τ_j affects the willingness to interact ($w_i(t) = 0$). Two factors are considered, the equipment required by τ_j and the distance d to the task. The case in which agent a_i does not have the necessary equipment required by task τ_j, will reflect in a negative willingness to interact. It is possible to distinguish two circumstances in which an agent a_i considers the allocation of τ_j, (i) a_i has no previous allocation, and (ii) a_i is already allocated to a set of tasks. In case (i), d is the distance between the agent's location and τ_j's location, with utility calculated as:

$$u_{\tau_j}(t) = 1/d. \tag{9.2}$$

In case (ii), d is the minimum distance to τ_j considering the a_i's location, and the location of the other tasks allocated to a_i, given by:

$$d = min(\{d_{kj}, \forall k \in L\}), \tag{9.3}$$

where L is the set containing the locations of agent a_i and its tasks, and d_{kj} is the distance between the k^{th} element in L and task τ_j. The final value of the willingness to interact with respect to task τ_j is expressed by

$$w_{i\tau_j}(t) = w_i(t) + u_{\tau_j}(t). \tag{9.4}$$

Although the willingness to interact is itself an expression of utility, in this paper its notion and that of task utility are separated. This is done in order to have a clear distinction between what affects the general disposition to collaborate, and what affects the utility of a performing a single task.

9.4.3 Interaction Protocol

Several assumptions hold concerning the interaction between agents. Firstly, no two agents can start the negotiation for a unique task at the same time. Secondly, agents can have the knowledge of each other's allocations, as well as the tasks that are completed. Thirdly, this knowledge is not necessarily available for every time-step, and can come to the knowledge of an agent with

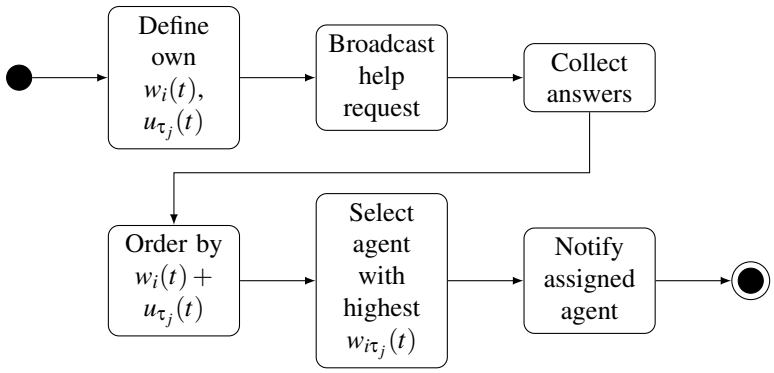


Figure 9.2: Interaction Protocol.

a certain delay. As a consequence of the last two assumptions it can happen that a task is repeated more than once.

The interaction protocol defines how agents negotiate with one another over the assignment of tasks which are to be completed (Fig. 9.2). Requests for help can be initiated by any agent in the event of the detection of a full failure of an agent [4]. The first agent to detect the failure of another agent initiates a negotiation with others to re-allocate the tasks of the failed agent. For each task to be assigned, a request for help is broadcast. Afterwards, the responses of other agents, consisting of the respective willingness and utility values, are collected. Note that, replies from agents with negative willingness are ignored. Thereafter, the rest of the responses are ordered based on the combined value of willingness and utility, and the agent with the highest willingness will be allocated to the task.

9.5 Centralized Global Planner

The process of mission planning first starts with the creation of a mission. This is done by a human operator who defines the mission parameters (tasks to be done, available vehicles and the overall goal of the mission) in the Mission Management Tool (MMT) [15]. After this step, the mission is sent to the planner which solves the mission and produces the necessary set of actions (plan)

[4] Such detection mechanisms are outside the scope of this paper. Further details on the implementation are given in Section 9.6.

for mission execution.

Algorithms used to solve this kind of problems are usually divided into two groups, exact and meta-heuristic. While exact algorithms can guarantee that the produced solution is optimal, meta-heuristics usually have no guarantees at all. However, meta-heuristic algorithms can produce a reasonably good solution within a short period of time. This is sometimes more important than having an optimal plan, especially in situations where re-planning might be necessary. Although the initial plan making is not bounded by time, the re-planning is. Re-planning, in this case, can be seen as planning again with new initial conditions. Since multi-agent missions are usually costly and autonomy of agents is limited as well, the re-planning process should be very fast. For that reason, the algorithm behind the global planner, in this paper, is a Genetic Algorithm (GA), which is adapted to planning and scheduling problems. Chromosomes are encoded in the same way as in [15], thus two arrays of integers, representing the genes, are used. The first array consists of integers representing tasks and agents, whereas the second array represents task parameters (equipment requirements, task duration, and location). Chromosome length varies from $n+1$ to a maximum of $n+m$ genes, depending on the number of agents used in a mission.

The initial population is randomly created with the respect to given constraints, hence, initial candidate solutions are in the feasible region of the search space.

The crossover operator has not been used since it did not have positive effects on the convergence process. Mutation is the only source of variability as it allows genetic diversity in the population. Every individual has a low probability to be selected for mutation. In this paper, two types of mutation schemes are introduced. One operates on the task genes through swapping tasks and inserting new genes, whereas the other mutates agent genes through growing (adding agents) and shrinking (removing agents) from the chromosome.

A task swap mutation swaps two task genes in a chromosome, meaning that it can both swap tasks within a single agent or between two agents. An insert mutation chooses a task and inserts it in a new location in a chromosome, similarly to the previously explained mutation, the insertion can be within the same agent or different one.

An agent shrink mutation removes one agent from a chromosome, reallocating its tasks to other agents. Growth agent mutation adds a new agent to the plan. The new agent gene is randomly inserted, acquiring tasks from that location in the chromosome up to the next agent gene or end of the chromosome. If there are conflicting (a task not supported by assigned agent) tasks,

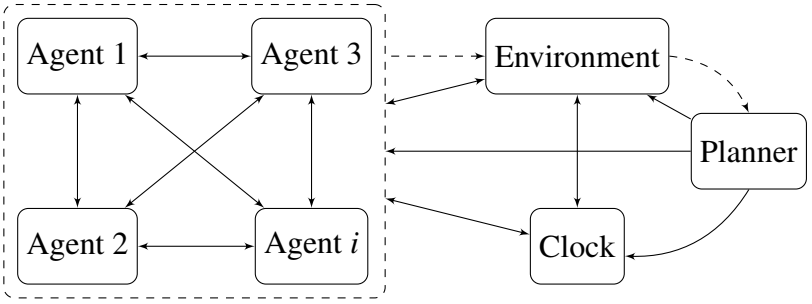


Figure 9.3: Simulation components.

they are randomly reallocated to other agents. Both algorithms take into account given constraints ensuring that the mutation process does not produce infeasible solutions.

9.6 Simulation Design

The combined approach consisting of the integration of a centralized planner and decentralized agents was compared to the planner-only approach. Both approaches were evaluated and compared in simulations. This section initially provides a general description of the simulation design, followed by implementation details concerning each of the approaches separately.

9.6.1 Simulation Design

The agent simulation (Fig. 9.3) is built on top of the ROS (robot operating system) middle-ware [18]. Agents, called nodes in ROS terminology, are of two types, (i) operative agents described in Sect. 9.4, and (ii) special purpose agents such as the clock and the environment agents.

Operative agents have the goal of completing the tasks which are part of the mission. Agents are heterogeneous with respect to the set of capabilities they have. In addition, more than one agent can have the same capability and one agent can have more than one capability. Furthermore, they are homogeneous with respect to the implemented motion model with a maximum velocity v_{max} set to $10m/s$. Their behaviour in the *idle* state, i.e., remaining stationary, is also the same. Agents communicate with one another through the publish/subscribe broadcast mechanism, and are able to listen to messages from each other, i.e.,

without a limitation in range. When an agent makes a request, it will wait until the first replies arrive or until a specified timeout of 5 seconds has passed. Consequently, not all responses from all other operative agents are necessarily considered. During the course of the simulation, any such operative agent can experience a full failure (determined in the environment node), which means that it has broken down and cannot do any task, or communicate with others.

The clock agent keeps track of the simulation time. After a simulation is initiated, the clock starts ticking when a plan has arrived from the planner. This is to ensure that across different runs of the simulations, the arrival time of the plan is the same and deterministic. The clock tick count is increased by 1 time unit every time all non broken down operative agents and the environment node have updated their state once. Also, it is assumed that any interaction between operative agents happens within the same time-step, i.e., the clock stops ticking when they are interacting, and resumes once the interaction has finished. The communication with the clock agent is realized through ROS one-to-one service calls.

The environment agent is used for three purposes: to keep track of information that concerns all agents, to determine which agent will fail at a given time-step, and as a locking mechanism. With respect to the first purpose, the information collected by the environment consists of the locations of agents and tasks in a $2D$-space, as well as lists of allocated and completed tasks by every operative agent. After the data is collected, it is broadcast as a whole – through the ROS publish-subscribe mechanism – to all operative agents, with the update taking place at every time-step. As a result, agents are aware of how tasks are allocated, and which tasks are completed at every time-step. Delays of a couple of time-steps might occur if some messages are lost or not received in time.

The second purpose of the environment is to simulate both the failure of an agent in the system, and the detection of such failure by other agents. Regarding the former, the environment initially calculates the time-step t_f in which to inject the failure as follows:

$$t_f = \text{randint}(0.2 \cdot m_D, 0.8 \cdot m_D) + t_{rp}, \tag{9.5}$$

where the function randint generates a random integer that lies between the two given arguments, m_D is the estimated mission duration, and t_{rp} is the time the previous re-plan has taken place, whether by the planner or the agents. Additionally, the environment randomly selects an agent to fail from a list of agents that are currently executing. Regarding the latter, after the occurrence of a failure is simulated, the environment node informs all agents, one by one,

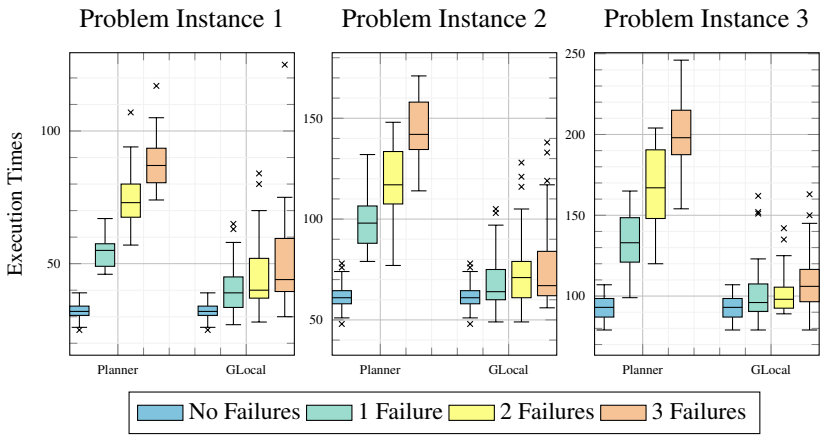


Figure 9.4: Execution times for both planner and hybrid approach across problem instances 1-3.

through ROS one-to-one service calls. Note that, if an operative agent completes a task at time t_f, there is no time for such information to propagate to the rest. As a result, such a task will be reallocated and its execution will be repeated by another agent. The third purpose of the environment is to lock the tasks and the invocation of the planner. This means that, (i) two agents will not be able to initiate a negotiation for the same task at the same time-step, and (ii) agents cannot contact the planner after a call has already been placed and before a new plan has been received. In the meantime, operative agents drop their assigned tasks and change the state to *idle*. In order to lock, either the tasks or calls to the planner, agents communicate with the environment through ROS one-to-one service calls.

9.6.2 Simulation Scenarios

Every mission starts with a human operator defining a set of tasks to be executed in the MMT. After mission creation, it is forwarded to the automated high-level planner described in the Sect. 9.5, where it is translated into ECTSP model [15]. The result of the planning process is a plan that is then forwarded to agents for execution. In both scenarios, the planner initiates the agent simulation by sending a plan with tasks allocated to agents and other mission relevant information such as the initial locations for agents and tasks, the required

equipment, the task duration, and the physical capabilities and limitations of involved agents.

The planner-only approach scenario In this approach, the planning and re-planning is only performed by the centralized planner. After the initial plan is generated, the planner goes into the idle state, waiting for re-planning requests, whereas agents begin executing the assigned tasks. When a failure is detected, the first agent that is able to lock the planner through the environment node, will immediately issue a re-planning request. Noticeably, there is no collaboration between operative agents. While waiting for the new plan, the clock node keeps ticking and all agents switch to the *idle* state. When the new plan arrives, t_{rp} is set to the current time t_n. It is important to emphasize that failures are induced in such a way that they cannot make a mission infeasible, i.e., the re-planning process will always be able to produce a feasible plan.

The GA planner is configured as follows. The population size is fixed to 500, and the number of generations is limited to 5000. The crossover operator was omitted, and the mutation probability is set to 10%, In addition, elitism is set to 5% in order to preserve the best candidate solutions from the previous generation.

The GLocal approach scenario In this approach, the planner and agent approaches are combined, i.e., agents are able to collaborate with one another and re-allocate tasks in the case of a failure during mission execution. As in the previous approach, after the initial plan, the planner goes to the idle state, while the agents begin the execution of the assigned tasks. When a failure is detected, all other operative agents compute the list of tasks $\mathcal{V}_x$ assigned to the failed agent x, removing those tasks perceived as complete. Then, a negotiation round begins for every task $v \in \mathcal{V}_x$. The first agent that manages to lock the task through the environment node, will initiate the corresponding re-allocation, by sending out help requests to all other agents. An agent will wait for a maximum time of $t_w = 5$ seconds [5] for any replies, or until the some replies have arrived, i.e., if the list of replies is not empty when an agent makes the check, then the waiting stops, and the agent proceeds with the negotiation using the list of replies available at that moment. Thereafter, it will order the replies in descending order of the willingness to interact value, and will assign the task

[5] Note that, the length of the timeout is chosen arbitrarily. Depending on the needs of the application, a shorter or longer timeout could be selected. It is assumed that for the purposes of this paper $t_w = 5$ seconds is an adequate upper bound.

to the agent that has the highest value. In case the agent has the necessary equipment for completing the task, then it will itself be part of the candidates that could be assigned to the task. In case such negotiation fails, e.g., the agent does not get any positive replies, and if the agent itself is not capable of performing the task, then a re-planning request will be issued. This request will invoke the centralized planner with the updated world state information. Note that, one failed negotiation is sufficient for triggering a call to the centralized planner. On the other hand, if agents manage to re-allocate all the tasks by themselves, the planner is not engaged. In case the planner is called after an agent failure, then t_{rp} is set to the time of the arrival of the new plan. Otherwise, t_{rp} is calculated as $t_f + 3$, as it is assumed that most agents will have received such information within the 3 time-steps. A re-allocation that happens as a result of the negotiation between agents takes place within a t_f time-step. Three more extra time-steps are added to give enough time to agents to propagate the new information pertaining the newly allocated tasks, before a new failure is introduced.

9.7 Results

The results of the comparison of the two aforementioned approaches are presented in this section. In order to gain an insight into the differences between the approaches, a series of statistical tests were conducted. We formulate the **null hypothesis** as *"A hybrid approach that combines a centralized and decentralized planner has no significant effect to the overall results compared to a centralized planner-only approach."* The data used for the comparison consist of execution times per each of 30 runs for every test case. The sample data has no normal distribution, i.e., data can be highly skewed and can have extended tail. This is the reason why the median value was chosen over the mean value and consequently the non-parametric Wilcoxon rank sum test is used. The results of the test are shown in the Table 9.1. Based on the results we can reject the null hypothesis since it is clear that the hybrid approach performs better with statistical significance. The agent only approach has been omitted from this comparison since it is generally accepted that decentralized planning results in worse sub-optimal plans compared to centralized planning in failure-free, non time-critical scenarios. This applies to the production of the initial plan in presented scenarios.

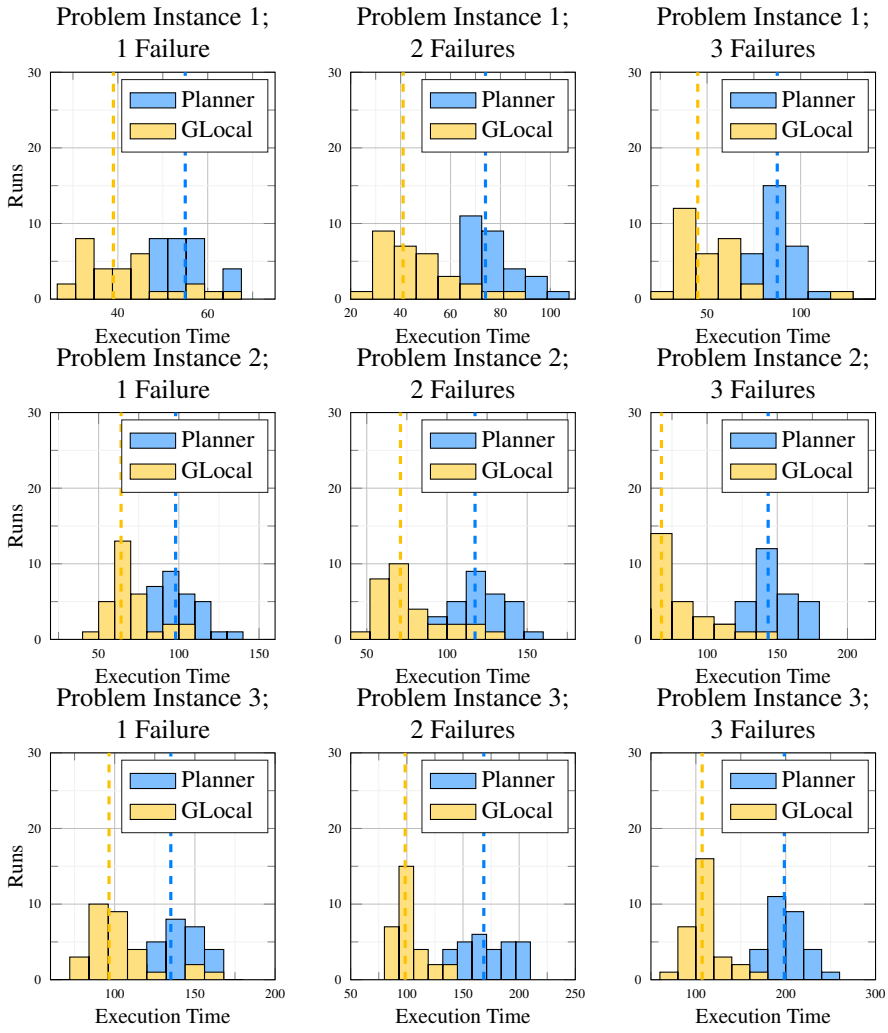


Figure 9.5: The distribution of the execution times for each of the three failures in each problem instance.

Table 9.1: Wilcoxon rank sum test applied on results obtained by planner only and hybrid approach.

	Problem Instance 1			**Problem Instance 2**			**Problem Instance 3**		
	1 failure	2 failures	3 failures	1 failure	2 failures	3 failures	1 failure	2 failures	3 failures
p-Value	$1.3 \cdot 10^{-6}$	$1.2 \cdot 10^{-8}$	$6.1 \cdot 10^{-10}$	$2.3 \cdot 10^{-8}$	$1.4 \cdot 10^{-8}$	$2.1 \cdot 10^{-10}$	$6.5 \cdot 10^{-7}$	$6.6 \cdot 10^{-11}$	$3.3 \cdot 10^{-11}$

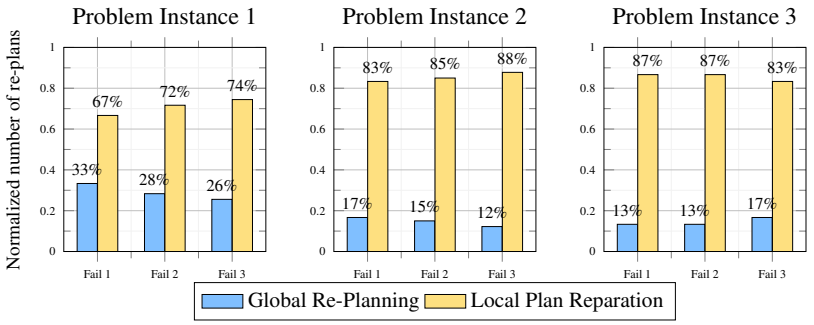


Figure 9.6: Normalized cumulative number of re-plans over 30 runs for each problem instance.

Benchmark Settings There are four benchmark settings which differ on the number of agent failures f that can happen in the system, $f \in \{0,1,2,3\}$. Each settings has 3 different problem instances varying in the number of tasks $n_T \in \{50,100,150\}$, where n_A is the number of agents. In total 36 distinct simulations are run. The number of repetitions for each case is $n_R = 30$. The number of agents in the system is fixed to $n_A = 10$. A failure cannot make the completion of the mission infeasible, i.e., there will always be at least one non-broken agent with the necessary equipment needed by any task. Additionally, the simulations run until all tasks have been completed. Therefore, all tasks will eventually be completed by the agents.

Scenario Comparison The average execution times across the different number of failures, task instances, and the two approaches are given in Fig. 9.4. It can be observed that with the increase of the number of failures, the mission makespan increases as well in the planner-only approach. Whereas in the hybrid approach, due to the plan reparation performed by the agents, the makespan is on average lower than in the planner approach. Additionally, as the size of the problem instance increases, the average disparity between the two approaches also increases in most cases. Similarly, as the number of the failures increases in a problem instance, the p-value decreases (Tab. 9.1). The distribution of the execution times for each of the test cases can be seen in Fig. 9.5. The dashed lines represent median values. It can be observed that the median value of the hybrid approach is always better (in this measure lower is better) than the median value of the planner-only approach. Although, some of

the solutions overlap, it is clear that the hybrid approach, in general, produces better solutions than the planner-only approach.

For the hybrid approach, the cumulative number of re-plans by the planner and the agents for each failure case across the three problem instances is calculated, and given in Fig. 9.6. Note that the total number of re-plans in a simulation run is equal to the number of failures f. As a result, the total number of re-plans can be calculated as $f \cdot n_R = \{0, 30, 60, 90\}$. Additionally, by inspecting Figs. 9.4 and 9.6, it is possible to note the influence of the number of times agents locally repair the plan over the execution times. In problem instance 1, failure case $f = 1$, agents do not invoke the planner 67% of the time a re-plan is needed (Fig. 9.6), thus impacting the variability of the execution times, which partly overlaps the results gained with the planner-only (Fig. 9.4). It can be noted that, depending on the problem instance, when a re-plan is needed the local plan reparation technique, involving only agents, is able to overcome the failure in the system and re-allocate tasks from the failed agent in 67–88% of the cases.

9.8 Related Work

To the best of authors' knowledge, only two works aim at addressing the problem of combining centralized and decentralized planning approaches in order to solve multi-robot task allocation problems. Le Pape has argued for the necessity of combining centralized and decentralized planning approaches to deal with uncertainties and unforeseeable events in dynamic environments [20]. Agents were given the option to create their own individual plans in a decentralized way. However, in the case of addition of new tasks into the system, e.g., as a result of the action taken by a human operator, agents invoke the centralized planner to make the task allocation of newly added tasks. Thereafter, agents proceed with the execution of the tasks. Duan *et al.* consider the online dispatching problem for dynamic autonomous taxi operations, the aim of which is to assign requests to the taxis in the system, while maintaining the feasibility of existing routes [21]. Requests can arrive at any time, and the time aspect is divided into short-term and long-term horizons. An immediate request is part of the short-term horizon, and is dealt with by a centralized planner that makes the assignment. Reservation requests, on the other hand, belong to the long-term horizon, and are dealt with by the autonomous taxis. In this approach, the taxis integrate these requests such that their routes remain feasible, until they are part of the short-term horizon and the planner makes the assignment.

Table 9.2: Using the TAMER model [19] to classify the problems addressed in combined centralized and decentralized planning.

TAMER / Ref.	Entities: Robot	Env	Mission	Task	Relationships: Team.	Comm. with	Depl.	Incl. (R&T)	Alloc.	Dep. on	Deco. of	Approach
Le Pape [20]	finite number of sensors; can determine action outcome; can plan; can generate goals	uncertainty (open);	tasks; capability constraints	single-robot task; capability needed (mobility)	–	communication with planner; communication between agents; asynchronous communication	Graph	*n* heterogeneous R;	minimize plan duration; wait or plan with current knowledge; 2 planning levels (planner & robot); time-extended	precedence	divisible into actions	Human/agent give planner a new goal; planner informs available agents; agents create individual plans for achieving the goal; planner collects the proposals and makes the final allocation; agents execute in a decentralized way
Duan *et al.* [21]	state (availability, location); able to estimate travel times; single-task robot	uncertainty (closed); discrete;	tasks; timing constraints;	single-robot task;	–	communication with: planner, clients, and other taxis	Graph	*n* functionally homogeneous R; constantly incoming T (requests)	minimize plan duration; minimize disjoint paths for traversing all nodes; time-extended	timing	atomic	Planner deals with requests in the short-term horizon; agents incorporate reservations far in the future in the long-term horizon, in order to maintain route feasibility;
Proposed Approach	state (location); finite number of capabilities; adaptive autonomy; determine who is broken; generate own goals	uncertainty (open); observability (fully); deterministic; discrete (with respect to time);	tasks; capability constraints; number of robots constraints	single-robot task; equipment needed	teams of size 1; no hierarchy between agents	broadcast	2D space	*n* heterogeneous R; *m* heterogeneous T	minimize plan duration; time-extended	no dependencies	atomic	Plan initially issued by the centralized planner; agents attempt execution; on failure agents attempt to repair the plan; in case of non-success in re-allocation agents ask for a re-plan.

Their goal is to reduce the planner workload from requests that are too far in the future.

A more detailed comparison, from the problem definition viewpoint, is provided by expressing the problems addressed in related works, as well as the problem tackled in this paper, through the TAMER model proposed by Miloradović *et al.* [19] (Tab. 9.2). TAMER is an entity relationship model that characterizes multi-robot allocation problems through four entities – *Robot*, *Environment*, *Mission*, *Task* and the relationships between these entities such as *teamed* (Team.), *communicate with* (Comm. with), *deployed* (Depl.), *includes robots* (Incl. R.), *includes tasks* (Incl. T.), *allocation* (Alloc.), *depend on* (Dep. on), and *decomposed off* (Dec. off) [6] *Teamed* and *Communicate with* capture the relationship between instances of the *Robot* entity, in terms of how robots collaborate with one another, and lower level concerns for communication, e.g., bandwidth, range etc. A *Mission* instance is *deployed* in an instance of Environment. Additionally, a *Mission includes* a set of *Robot* and *Task* entities. *Task* entities are connected with one another through the *depend on* and *decomposed off* relationships. Finally, the *Robot*, *Task*, and *Mission* entities are linked through the *allocation* relationship that captures properties such as the allocation type (instantaneous assignment versus time-extended) and utility function.

9.9 Conclusion

In this work, a hybrid approach for multi-agent mission re-planning is proposed. This approach combines high-level global planning realized by a centralized planner, with a local plan reparation technique carried out in a decentralized manner by a group of agents. In the event of full failures, i.e., agent breakdowns, the remaining agents attempt to repair the plan locally by re-allocating any pending tasks among each other. If at least one task remains un-allocated, agents invoke a centralized planner which generates a new plan, for all agents and unfinished tasks, based on the updated information received from the agents. The hybrid approach is compared to a planner-only approach in simulations, and it is shown that on average it achieved better results, i.e., shorter mission make-span as compared to the latter, in the presence of different numbers of failures.

There are three lines of inquiry for future work. First, it is of interest to

[6] The *act* relationship binding the *Robot* and *Environment* entities has been omitted from Table 9.2, due to no specification in the given works.

investigate the performance of the hybrid approach in the presence of partial failures of agents. A partial failure could mean a faulty equipment for an agent that prevents the execution of some of the tasks allocated to the agent. Second, the proposed approach – with additional local strategies by agents for an efficient local re-planning – can be applied to solve the Extended Colored Traveling Salesperson Problem (ECTSP), with precedence and/or synchronization constraints among others. Finally, different strategies for determining the re-planning threshold, i.e., when to call the planner, are to be investigated.

Bibliography

[1] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[2] Alejandro Torreño, Eva Onaindia, Antonín Komenda, and Michal Štolba. Cooperative multi-agent planning: A survey. *ACM Comput. Surv.*, 50(6):84:1–84:32, November 2017.

[3] Heiko Hamann. *Swarm Robotics: A Formal Approach*. Springer International Publishing, 2018.

[4] R. Stanković, M. Štula, and J. Maras. Evaluating fault tolerance approaches in multi-agent systems. *Auton Agent Multi-Agent Syst*, 31:151–177, 2017.

[5] Giovanni Beltrame, Ettore Merlo, Jacopo Panerati, and Carlo Pinciroli. Engineering safety in swarm robotics. In *Proceedings of the 1st International Workshop on Robotics Software Engineering*, pages 36–39. ACM, 2018.

[6] Diana Spears, Wesley Kerr, and William Spears. Safety and security multi-agent systems. In *Safety and Security in Multiagent Systems*, pages 175–190, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[7] Thalia May Laing, Siaw-Lynn Ng, Allan Tomlinson, and Keith M Martin. Security in swarm robotics. In *Handbook of Research on Design, Control, and Modeling of Swarm Robotics*, pages 42–66. IGI Global, 2016.

[8] Yaqin Hedin and Esmiralda Moradian. Security in multi-agent systems. *Procedia Computer Science*, 60:1604–1612, 2015.

[9] Marie E. desJardins, Edmund H Durfee, Charles L Ortiz Jr, and Michael J Wolverton. A survey of research in distributed, continual planning. *AI magazine*, 20(4):13–13, 1999.

[10] Mathijs De Weerdt and Brad Clement. Introduction to planning in multi-agent systems. *Multiagent and Grid Systems*, 5(4):345–355, 2009.

[11] Alejandro Torreño, Eva Onaindia, Antonín Komenda, and Michal Štolba. Cooperative multi-agent planning: A survey. *ACM Computing Surveys (CSUR)*, 50(6):1–32, 2017.

[12] Itziar Landa-Torres, Diana Manjarres, Sonia Bilbao, and Javier Del Ser. Underwater robot task planning using multi-objective meta-heuristics. *Sensors*, 17(4):762, 2017.

[13] Cristian Ramirez-Atencia, Gema Bello-Orgaz, María D. R-Moreno, and David Camacho. Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms. *Soft Computing*, 21(17):4883–4900, 2017.

[14] S. Karaman, T. Shima, and E. Frazzoli. A process algebra genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 16(4):489–503, 2012.

[15] Branko Miloradović, Baran Çürüklü, Mikael Ekström, and Alessandro V Papadopoulos. A genetic algorithm approach to multi-agent mission planning problems. In *International Conference on Operations Research and Enterprise Systems*, pages 109–134. Springer, 2019.

[16] Abderraouf Maoudj, Brahim Bouzouia, Abdelfetah Hentout, and Redouane Toumi. Multi-agent approach for task allocation and scheduling in cooperative heterogeneous multi-robot team: Simulation results. In *IEEE Int. Conf. on Industrial Informatics (INDIN)*, pages 179–184, 2015.

[17] Mirgita Frasheri, Baran Cürüklü, Mikael Esktröm, and Alessandro Vittorio Papadopoulos. Adaptive autonomy in a search and rescue scenario. In *2018 IEEE 12th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 150–155. IEEE, 2018.

[18] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009.

[19] Branko Miloradović, Mirgita Frasheri, Baran Cürüklü, Mikael Ekström, and Alessandro Vittorio Papadopoulos. TAMER: Task allocation in multi-robot systems through an entity-relationship model. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 478–486. Springer, 2019.

[20] Claude Le Pape. A combination of centralized and distributed methods for multi-agent planning and scheduling. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 488–493. IEEE, 1990.

[21] Leyi Duan, Yuguang Wei, Jinchuan Zhang, and Yang Xia. Centralized and decentralized autonomous dispatching strategy for dynamic autonomous taxi operation in hybrid request mode. *Transportation Research Part C: Emerging Technologies*, 111:397–420, 2020.

Chapter 10

Modeling the Willingness to Interact in Cooperative Multi-Robot Systems

Authors: Mirgita Frasheri, Lukas Esterle, Alessandro Vittorio Papadopoulos
Venue: 12^{th} International Conference on Agents and Artificial Intelligence (ICAART'20)

Abstract

When multiple robots are required to collaborate in order to accomplish a specific task, they need to be coordinated in order to operate efficiently. To allow for scalability and robustness, we propose a novel distributed approach performed by autonomous robots based on their willingness to interact with each other. This willingness, based on their individual state, is used to inform a decision process of whether or not to interact with other robots within the environment. We study this new mechanism to form coalitions in the on-line multi-object κ-coverage problem, and compare it with six other methods from the literature. We investigate the trade-off between the number of robots available and the number of potential targets in the environment. We show that the proposed method is able to provide comparable performance to the best method in the case of static targets, and to achieve a higher level of coverage with respect to the other methods in the case of mobile targets.

10.1 Introduction

Robots collaborating with each other in order to tackle a task perform faster and more efficiently than their individually operating counterpart. Some tasks even require collaboration of multiple robots and cannot be accomplished by individual robots at all. However, such collaboration requires coordination of the individual robots, and formation of coalitions between them. Numerous coalition formation approaches have been proposed which either rely on central components [1–3] or focus on a single task to be accomplished [4] in order to achieve meaningful interaction and collaboration. These approaches require dissipation of information about available coalitions as well as negotiations about participation of each potential coalition member [3–5]. While coalitions are usually formed around single tasks, the use of multiple teams has been shown to be beneficial when pursuing goals that require multiple tasks to be accomplished concurrently [6, 7]. Moreover, when considering autonomously operating robots that aim to achieve multiple tasks, the individuals have to make decisions on when and how to form coalitions, and to what end the coalition is formed.

In this work, we are interested in the ability of autonomously operating robots to interact and collaborate in order to provision varying sets of tasks efficiently, without a central component involved. We propose an approach where each robot makes individual decisions about whether or not to provision a specific task, employing local information about its own status, e.g., its battery level, its ability, and its interest (i.e., expected performance value the robot contributes to the collective) in performing such task. More specifically, we propose a novel distributed coalition formation and study this approach in the online multi-object κ-coverage problem [8, 9], which is related to the cooperative multi-robot observation of multiple moving targets (CMOMMT) problem proposed by Parker and Emmons [10], and consists of a varying number of tasks required to be tackled concurrently.

First, the robots need to discover initially unknown moving objects in the environment. They do not possess any *a priori* information about the number or location of these objects. Furthermore, objects may be mobile, requiring robots to change their own location respectively in order to continuously provision them. Second, each object needs to be provisioned with at least κ robots concurrently, i.e., κ robots having the object within their sensing/actuating region at the same time. Here, detecting new targets is considered the first task, however, every newly discovered target generates a new task for the collective of covering this known target. This generates a trade-off between detecting new

objects and covering known objects with κ robots when the collective tries to maximize the duration and number of targets covered by κ robots. However, a robot not only needs to decide between provisioning a specific target or exploring the area to discover new targets, but also which of the different known targets it wants to provision. In order to achieve an efficient outcome in this trade-off, the robots are required to form new coalitions for each individual target. According to the taxonomy of Robin and Lacroix [11], the on-line multi-objective κ-coverage problem is hunting mobile search, monitoring multiple targets, with different viewpoints.

In this paper, we present a novel distributed coalition formation algorithm considering several tasks. At its core, we propose to introduce a *willingness to interact* to each individual robot as the main driver for the coalition formation. The *willingness* is dependent on the state of the robot, such as local conditions like battery level, and current level of activity. Utilizing this *willingness*, robots can make decisions on whether or not to interact and provision a specific object which eventually leads to forming coalitions with other robots. This approach is evaluated over several scenarios of increasing number of targets, considering both static and mobile targets separately. The performance is assessed through different metrics, e.g., the average number of agents covering one target, the average coverage time with at least κ agents. Furthermore, the proposed approach is compared against six other methods presented in the literature. The proposed approach shows performance that is either comparable with the best of the methods it is compared with – in the case of static objects – while it exhibits a higher coverage in the case of moving targets.

The remainder of this paper is structured as follows. Section 10.2 gives a formal definition of the online multi-object κ-coverage problem and Section 10.3 covers the behaviour of the agents and targets, their interaction as well as our novel coalition formation algorithm. Section 10.4 gives an overview of the experimental setup, the performed experiments, and the obtained results. Section 10.5 discusses the generalization of the proposed approach, while Section 10.6 concludes the paper and outlines future work.

10.2 Problem Formulation

In the online multi-object κ-coverage problem, we assume a discrete 2D area Z with a given width and height w and h, respectively, without any obstacles. We also consider a set of active robots $A = \{a_1, a_2, \ldots, a_n\}$, and a set of targets or objects of interest $O = \{o_1, o_2, \ldots, o_m\}$ in this problem. Both robots and objects

can freely move within Z, with (nonconstant, yet limited) velocities v_i, where $i = 1,\ldots,n$, and v_j, where $j = 1,\ldots,m$; in their motion the robots will always remain in Z. It is assumed that any robot can move faster than the objects, and that the number of robots and targets is constant, i.e., targets cannot appear or disappear. Each robot is controlled by an internal, autonomous software agent. We refer to both as a_i. Each robot has a visibility range, with radius r. An object can be perceived by any robot, only if it is located within its visibility range. At this point, the robot will determine the number of already provisioning robots for this object. Therefore, it will either initiate a new coalition, in the case of no robots following the target, or join the existing coalition, in the case that less than κ agents are following the target. All objects are associated with a prescribed constant interest level l_j. Levels of interest are not necessarily the same between objects, and define the utility $u_{ij}(t)$ of a robot i for following an object j with interest level l_j at a discrete time-step t.

Every agent i can calculate its willingness w_i to interact with others (as detailed in Section 10.3.3) at each time-step. This can occur in different situations, e.g., (i) when a robot i first detects an object j entering or leaving its sensing area an object j is entering the sensing area of the robot i and (ii) when robot i receives an invitation to provision an object j from another robot. Robots are assumed to communicate with one another via broadcast, as implemented in ROS [12]. Thus, the willingness to interact shapes the cooperative behavior of an agent and its respective robot in relation to the others. Robots are able to change and keep track of their own state and behavior, as well as the state and behavior of other robots. Specifically, robot's n state is composed of the following variables: battery level b_i, range d, location $\ell_{x,y}$, and velocity $v_{a,i}$. Without loss of generality, we assume that the level of interest for the targets is robot-independent, i.e., there is a shared knowledge among the agents on the level of interest of different targets.

The online multi-object κ-assignment problem is solved by having at least κ robots covering any target in the set. Consequently, two tasks should be achieved concurrently: (i) maximizing the number of provisioned objects, and (ii) provisioning the targets with at least κ robots. This paper addresses the following questions:

1. What is the average time for which at least κ robots can cover all targets moving around in an environment when using the proposed coalition formation algorithm?

2. What is the average number of agents that can cover a target with the proposed coalition formation algorithm?

3. Is the motion of the targets affecting the obtained performance?
4. How does the defined value for κ affect the performance of the robot cooperation?
5. How does the proposed method compare with other state-of-the-art techniques for the κ-coverage problem?

We address these questions using two experimental setups with varying number of either mobile or static (immobile) targets, according to metrics that analyze the obtainable performance in terms of time to cover targets with at least κ robots, and the average number of robots that cover the targets. Furthermore, we compare our results with six other methods previously proposed in the literature [8].

10.3 Agent Model

In this section, we describe how a robot operates, how the agent, embodied in a robot, updates the willingness to interact, and how these agents form decisions to cooperate through the proposed interaction protocols. In the following we are using the terms robot and agent interchangeably.

10.3.1 Robot Kinematics

Every robot $a \in A$ follows a simple unicycle kinematic model

$$\begin{cases} \dot{x}_a(t) = v_a(t)\cos(\theta_a(t)) \\ \dot{y}_a(t) = v_a(t)\sin(\theta_a(t)) \\ \dot{\theta}_a(t) = \omega_a(t) \end{cases} \tag{10.1}$$

where $x_a(t)$ and $y_a(t)$ are the x- and y-coordinate on the map and define the position $\mathrm{p}_a = (x_a, y_a)$ of a robot a at time t, θ_a is the orientation of the robot, v_a is the forward velocity of the robot, and ω_a is its angular velocity. We assume that the robot can localize itself within the map, and that it can detect the obstacles within its visibility range.

A robot $a \in A$ follows a set of objects $O_a \subseteq O$, each of which has a different level of interest l. The direction d_a over which the robot moves is thus computed as

$$\mathrm{d}_a(t) = \frac{\sum_{i \in O_a} l_i(\mathrm{p}_a(t) - \mathrm{p}_i(t))}{\sum_{i \in O_a} l_i} \tag{10.2}$$

making the robot to move towards all the followed objects, weighted by their respective interest. In this way, the robot will prioritize targets with higher level of interest. The target orientation θ_a° and the forward velocity of the robot are therefore computed as:

$$\theta_a^\circ(t) = \angle \mathrm{d}_a(t), \tag{10.3}$$

$$\tilde{v}_a(t) = \|\mathrm{d}_a(t)\| \tag{10.4}$$

where $\angle \mathrm{p} \in [0, 2\pi)$ is the angle of the vector $\mathrm{p} = (p_x, p_y)$ in its reference frame, and it is obtained as $\angle \mathrm{p} = \mathrm{atan2}(p_y, p_x)$. In order to compute the proper value of the angular velocity, we can just use a simple proportional controller with tracking error e_a normalized between $[-\pi, \pi)$:

$$e_a(t) = \theta_a^\circ(t) - \theta_a(t) \tag{10.5}$$

$$\tilde{\omega}_a(t) = K_p \,\mathrm{atan2}\left(\sin(e_a(t)), \cos(e_a(t))\right) \tag{10.6}$$

Finally, we include saturations on the forward and angular velocities:

$$v_a(t) = \min(\tilde{v}(t), v_{\max}) \tag{10.7}$$

$$\omega_a(t) = \min(\max(\tilde{\omega}_a(t), -\omega_{\max}), \omega_{\max}) \tag{10.8}$$

10.3.2 Agent Behavior

Software agents, embodied in physical robots, operate autonomously and their behavior can be described as a state machine composed of four states: *inspect*, *evaluate*, *inspect & follow*, and *evaluate & follow*. Figure 10.1 shows the state machine that describes the behavior structure of an agent. At run-time, any agent starts its operation in the state *inspect*, in which it moves in Z according to a given pattern. In case a new target is spotted, or a request is received, an agent switches from *inspect* to *evaluate*. In the *evaluate* state, an agent decides how it wants to interact with the spotted target or the request for help, based on its current state. The proposed interaction protocol is described in detail in Section 10.3.4. The result of the interaction is a coalition of agents that will start following the spotted target. If the agent is not part of the coalition after the interaction, it will switch back to the *inspect* state, looking for other targets in the environment. Otherwise, if the agent is part of the coalition, then it will switch to the *inspect & follow* state. In this new state, the agent follows the target, while it simultaneously inspects for new ones. In case an agent loses track of all the targets it is following, then it switches to *inspect*. In case an agent has negative willingness, spots a new target, detects that a target is going

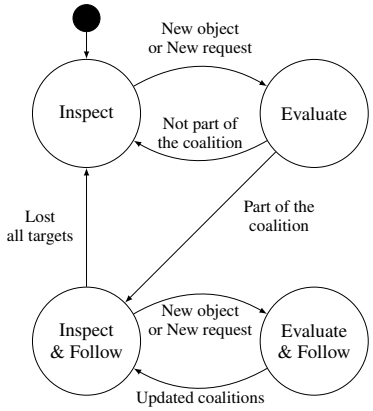


Figure 10.1: Agent operation state machine.

outside its visibility region, or it gets a request for help, then it switches to the *evaluate & follow* state. In this state, the agent either generates a help request, or it responds to a help request. In both cases, the agent decides if it will be part of a new coalition, or if it is going to drop a target. Once the interaction is complete, an agent switches back to the *inspect & follow* state with an updated set of targets to follow.

Note that an agent can be part of more than one coalition, i.e., can follow several targets simultaneously according to their level of interest (as per Eq. 10.2), but a target is only followed by a single coalition. Also, notice that transitions between states are considered to be instantaneous.

When an agent is following a set of targets, its motion is described by the dynamic model (10.1), and by control strategy defined in (10.3)–(10.8). The interest level of a target affects the motion of the agent according to (10.2), i.e., the agent's direction is mostly affected by the level of interest of the targets.

10.3.3 Willingness to Interact

The willingness to interact w shapes the cooperative behavior of an agent, i.e., when an agent should ask for help and when it should give help. This parameter does not refer to a particular task that should be completed, but rather reflects

the general disposition of any agent to cooperate with the others. The willingness to interact w takes values in $[-1,1]$. When $w \geq 0$ the agent is willing to provide support to other agents that have requested help. When $w < 0$, the agent will raise requests for help, and for $w = -1$ it cannot continue with the execution of a task on its own. The value of the willingness is updated by each individual agent i based on several individual factors, at discrete time instants t, according to the dynamics:

$$w_i(t+1) = \min(\max(w_i(t) + \mathrm{B}^\top \mathrm{f}(t), -1), 0), \tag{10.9}$$

$\mathrm{f}(\cdot) = [f_1(\cdot), \ldots, f_m(\cdot)]^\top$ is an $m \times 1$ vector of the m factors that affect the willingness, while $\mathrm{B} = [\beta_1, \ldots, \beta_m]^\top$ is an $m \times 1$ vector that contains the weights of the corresponding factors on the calculation of the willingness.

The calculation of a factor f_i is given by

$$f_i(k) = \phi_i(k) - \phi_{i,\min}, \tag{10.10}$$

where $\phi(k)$ represents the current measurement of that factor (e.g., the current battery level), while $\phi_{\min}$ is a minimal threshold considered acceptable (e.g., the minimal battery level to perform a task). The terms ϕ and $\phi_{\min}$ take values in $[0,1]$, where 0 is the minimum value of the measured quantity, and 1 its maximum.

In this work, we consider two factors that affect the willingness to interact. These are the battery level b, and the number of objects in O_a currently provisioned by a. Other factors can be included in the calculation of the willingness, without loss of generality of the proposed approach.

Factors can be divided into two categories: necessary and optional. The battery level is a necessary factor, since a robot with a battery level lower than a certain threshold may not be able to reach the moving target, or to complete an assigned task. Therefore, an agent with a low battery level should try to receive help from the other agents. On the other hand, the number of targets (n_O) an agent is tracking is considered as optional, since an agent can follow several targets, but this makes its task more difficult, e.g., if $1/n_O$ goes below a certain threshold – the agent is following too many targets – then the agent decreases its willingness to give help and consequently increase its willingness to ask for help. The effect of different factors is defined by their corresponding weights. The weight for a necessary factor β_{nec} is defined as:

$$\beta_{\text{nec}}(t) = \begin{cases} 1/m, & \phi_{\text{nec}}(t) - \phi_{\text{nec,min}} > 0, \\ -(1 + w(t)), & \text{otherwise}, \end{cases} \tag{10.11}$$

where m is the number of all the factors, whereas the weight for an optional factor β_{opt} is defined as:

$$\beta_{\text{opt}}(t) = \begin{cases} 0, \text{ if } \exists \phi_{\text{nec}}, \phi_{\text{nec}}(t) - \phi_{\text{nec,min}} < 0, \\ \dfrac{\text{sgn}(\phi_{\text{opt}}(t) - \phi_{\text{opt,min}})}{m}, \text{ otherwise.} \end{cases} \tag{10.12}$$

This ensures that necessary factors have the highest impact on the willingness to interact. As an example, in the case the battery level is below a threshold, then the agent should ask for help, irrespective of other factors ($w = -1$). Thus, the weights of other factors should be set to zero. While we provided an example for factors approaching a minimum, factors approaching a maximum can also be applicable. In such a case the calculation for factors and weights has to be adapted accordingly. More examples on factors that can affect the willingness can be found in [13].

10.3.4 Interaction Protocol

The interaction protocol defines how agents create coalitions for any given target and elect the corresponding leaders for these coalitions. The proposed protocol mostly complies with the SCR design pattern [14], however differently from SCR an agent can belong to different coalitions, hence it can have more than one leader. An agent can trigger a help request in case it spots a new target, or it wants to extend an existing coalition to reach $\kappa-$coverage, or it perceives that targets in its visibility range are moving away from itself, and if it is necessary to ask for help (e.g., battery level is under the accepted minimum). Furthermore, agents can decide to interact with one another when they receive help requests from others. The interaction protocol is illustrated in Figure 10.2.

When an agent spots a new target, it broadcasts an *information request* to other agents together with its willingness and respective utility for provisioning the target. The agent waits for a specified time Δt to receive a response from other robots. We assume that agents can identify commonly observed objects and assign common labels. In case a coalition exists already for the given target, the corresponding leader will reply whether or not further agents are needed to reach the κ-coverage. If no help is needed, then the agent continues its previous activities. If help is needed, then the agent will receive an assignment from the leader of the coalition, based on the previously sent willingness and utility. In case the agent does not receive a response within time Δt to its initial information request, it assumes no other agent is following the

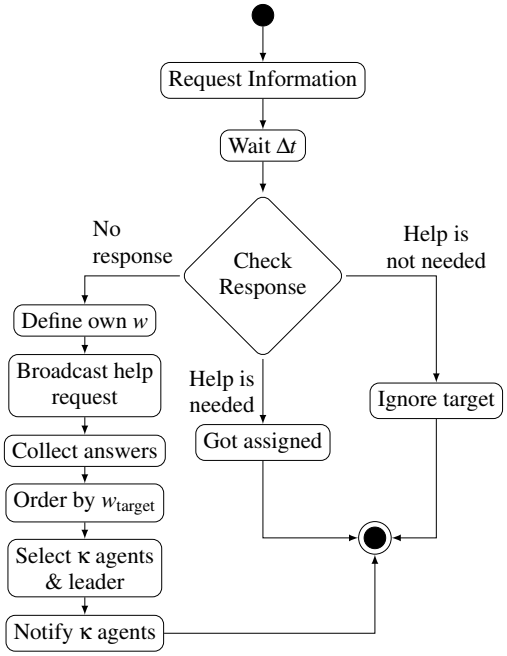


Figure 10.2: Activity diagram of the agent's behavior when a new target is spotted.

target. Subsequently, the process for creating a coalition and electing a leader responsible for following the target is triggered. Initially, the agent calculates its own willingness to help in the future coalition, and the utility from following the target. The mechanism follows the logic of a fast bully algorithm [15], well known in distributed systems. A request for help to follow the object is broadcast to all other agents. Other agents send their willingness to help, i.e, the willingness to enter the coalition, and their utility for following the specific target. After the responses are collected, agents with a negative willingness $w < 0$, are not considered further. Positive willingness of an agent i to interact is combined with its utility u_{ij} to form the willingness to interact to provision a specific object j at time t:

$$w_{ij}(t) = w_i(t) + u_{ij}(t). \tag{10.13}$$

Utilities are defined by each agent for the individual target and can generally vary between the different agents as well as the different targets. Examples for

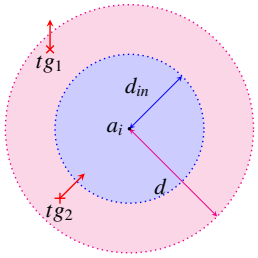


Figure 10.3: Agent a_i with visibility range d, indicated with the red circle, and internal range d_{in}, indicated with the blue circle. The targets tg_1 and tg_2 are indicated with crosses, and they are moving towards and away from the agent, respectively.

this could be the size, speed, or direction of movement of the object. In our experiments, we consider different interest levels that are agent-independent, i.e., the agents share the same interest for the same targets. The received values $w_{ij}(t)$ are ordered, and the κ agents with highest $w_{ij}(t)$ are selected for the coalition. The agent with the highest $w_{ij}(t)$ is elected the leader. The outcome is propagated to the other agents. The initiating agent does not necessarily need to be part of the coalition.

Furthermore, every agent keeps track of whether the targets in its visibility range are moving away from the robot. We introduce another internal threshold with radius d_{in} around the robot, where $d_{\text{in}} < d$ (Figure 10.3). When a target, e.g., tg_2 in Figure 10.3, moves out of the internal range, yet remains within the visibility range, then a request for help is triggered. If a target, e.g., tg_1, is moving towards the agent while being within the internal and visibility range, no request is issued. In case the willingness of an agent becomes negative, help requests are generated. At the same time, an agent will consider dropping its targets one by one. If the willingness remains negative or becomes -1, eventually all targets will be dropped.

A help request means that either an agent is looking for a replacement for itself, or it is looking for an additional agent that can enter the coalition. This is illustrated in Figure 10.4. If an agent needs to leave a coalition, we distinguish between leading agents and ordinary members of the coalition. If a leader agent needs to replace itself, then the leader election needs to be repeated. The process can include other agents not yet in the coalition, if κ-coverage is not achieved at that point in time. On the other hand, if a common agent needs to drop a target, then it first notifies its leader. Leaders are also responsible for triggering continuously the extension of a coalition in order to maintain

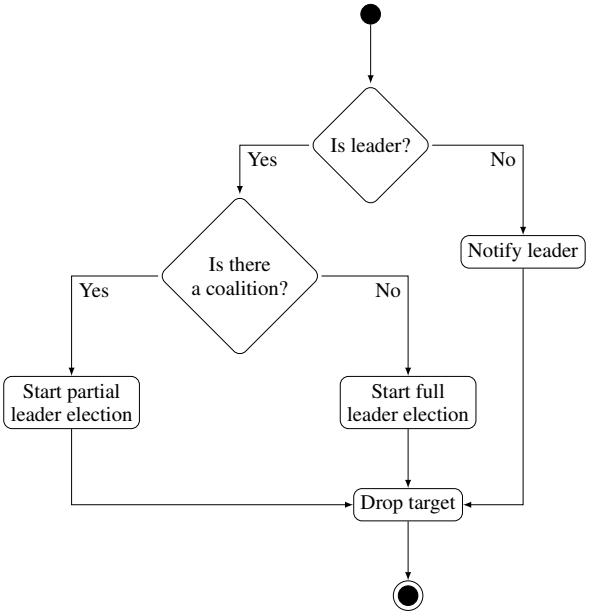


Figure 10.4: Activity diagram of the agent's behavior when it needs to replace itself in a coalition.

κ-coverage, following a monotonically increasing period.

10.4 Simulation Setup

The behavior of the agents was evaluated with computer simulations[1], based on the robot operating system (ROS) [12, 16] to model the agents kinematics, behavior, and interaction.

The method utilizing the willingness to interact, as described in this paper, was compared to six other methods that were previously proposed in the literature for solving the multi-object κ-coverage problem [8]. Each of the six methods is a combination of one communication model and one response model. Two communication models are considered, broadcast *BC* and random *RA*. In the broadcast model an agent broadcasts help request to everyone, whereas in

[1]The code for running the simulations is publicly available at https://gitagent@bitbucket.org/gitagent/gitagent_2.git

the random model it sends a help request to κ random agents. As for the response models, three are considered: (i) newest-nearest *NN*, (ii) available *AV*, and (iii) received calls *RE*. In the newest-nearest model an agent will answer to the request that is newest, and if there are multiple request at the same time, it will respond to the one that is nearest. In the available model the agent answers to requests according to the newest-nearest strategy only if it is not engaged in following other target(s). In received calls, an agent will answer to requests for objects with the least coverage, only if it is not following other targets. The six methods chosen for comparison are: *BC-NN*, *BC-AV*, *BC-RE*, *RA-NN*, *RA-AV*, and *RA-RE*. Such selection is due to previous results [8], where the broadcast and random communication models were evaluated better with respect to the rest, and the response models were reported to have a significant impact on the κ-coverage.

In all of our simulations, we consider a total number of $n_A = 10$ robots starting from the same initial position $(0,0)$, with a random direction, and $v_{i,\max} = 2$ units per time-step. If an agent hits any boundary in Z, it will bounce back at a 90° angle, i.e., we are considering a limited area surrounded by walls. The objects to be covered are distributed uniformly in the map Z of size $100\text{m}\times 100\text{m}$. We consider 7 different scenarios for our experiments with an increasing number of objects. The number of objects distributed in the environment are 1, 4, 7, 13, 16 and 19 for the corresponding scenarios *S*0 to *S*6. For each simulation the interest level of any target was randomly sampled from a set of levels $\mathcal{L} = \{0.3, 0.6, 0.9\}$. We performed 20 experiments for each scenario, with each experiment having a duration of $T_{\text{sim}} = 300$ discrete time steps and a specified seed. The latter impacts the initial location of the targets, the initial direction for agents and mobile targets, as well as the level of interest of targets for each experiment corresponding to a scenario. Given these settings, we analyzed the behavior of our agents to achieve κ-coverage where $\kappa \geq 3$, and $\kappa \geq 5$.

10.4.1 Results for Static Targets

In the first set of experiments, we consider only targets that remain in their initial location ($v_j = 0$). Once the targets are covered, they remain covered for the rest of the simulation, as such, the time for reaching the desired coverage is considered one of the performance indicators for evaluation.

For every scenario, we run N different experiments. For every experiment $e = 1, \ldots, N$, we compute for every target j the time to reach 1-coverage $t_{j,e}^{(1)}$, and the time to reach κ-coverage $t_{j,e}^{(\kappa)}$. Based on this information we can cal-

culate: (i) the average time to get one target to be covered by at least by κ agents $t_{\text{avg}}^{(\kappa)}$, and (ii) the average minimum time to get all the targets covered by at least κ agents $t_{\text{min}}^{(\kappa)}$. These two metrics give an indication of a minimum coverage, and a complete coverage, and the respective timing properties. They are formally defined as:

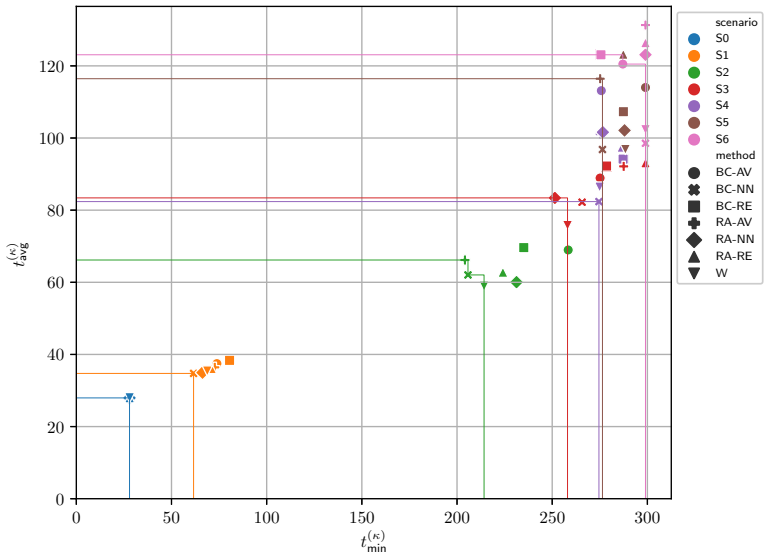
$$t_{\text{avg}}^{(\kappa)} = \frac{1}{N} \sum_{e} \frac{1}{|O|} \sum_{j} t_{j,e}^{(\kappa)} \tag{10.14}$$

$$t_{\text{min}}^{(\kappa)} = \frac{1}{N} \sum_{e} \max_{j} t_{j,e}^{(\kappa)} \tag{10.15}$$

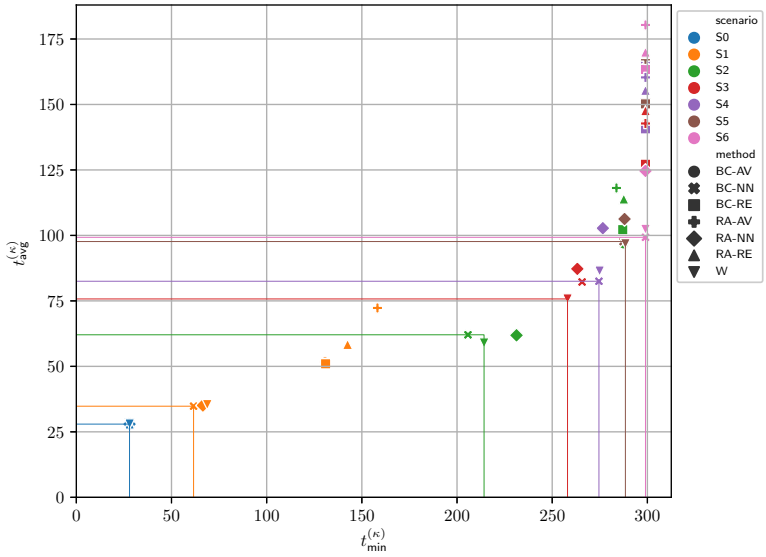
In particular, in these experiments we study (i) the average time to get one target to be covered by at least by 1 agent, $t_{\text{avg}}^{(1)}$, (ii) the average time to get one target to be covered by at least by κ agent, $t_{\text{avg}}^{(\kappa)}$, (iii) the average minimum time to get all the targets covered by at least 1 agent, $t_{\text{min}}^{(1)}$, and (iv) the average minimum time to get all the targets covered by at least κ agents, $t_{\text{min}}^{(\kappa)}$. In all the metrics, the lower, the better.

Results for $\kappa \geq 3$ as well as $\kappa \geq 5$ are shown in Figure 10.5, where $t_{\text{min}}^{(\kappa)}$ is given on the x-axis, and $t_{\text{avg}}^{(\kappa)}$ is given on the y-axis. In both metrics, the lowest value, the better. In the graph we also indicate the corresponding Pareto frontier to highlight the best performing methods. We also compare this directly to the cases for $\kappa \geq 1$ (only a single agent covers the target), however, the agents still aim to cover all targets with $\kappa \in \{3,5\}$ and therefore might cluster at specific objects even when reporting results for $\kappa \geq 1$.

It can be observed that on average there are no differences between the utilized methods for scenario *S*0 for $\kappa \geq 1$, as shown in Figure 10.5. This is due to the fact that there is only one static target in the environment, which will be discovered at the exact same time irrespective of the method for an experiment initiated with the same seed. There could be a shift with a couple of time-steps in the discovery times, in case there is an occasional failure in the ROS service calls or broadcast used by the agent when handling targets that appear in the visibility range. Nevertheless, for $\kappa \geq 5$, Figure 10.5d, the minimum times are not necessarily the same, e.g., the result for method *RA-AV* as compared to the six other methods.

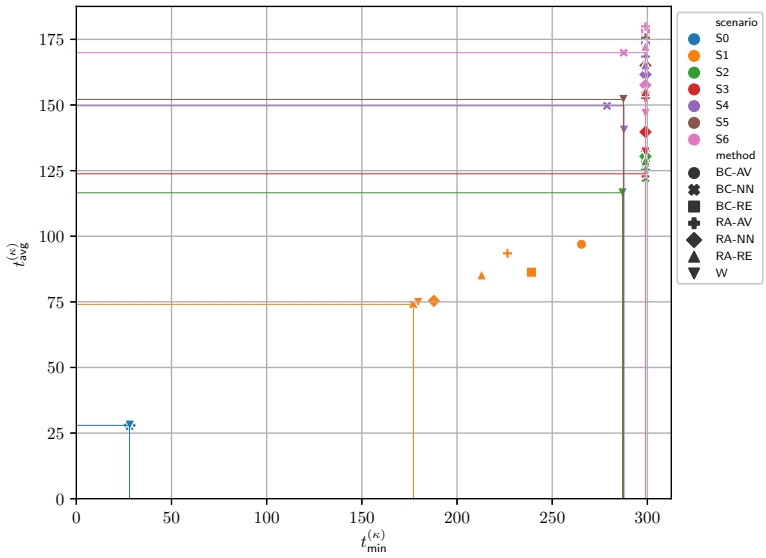


(a) Average vs Minimum Time to Coverage with at least one agent when tasked with $\kappa \geq 3$.

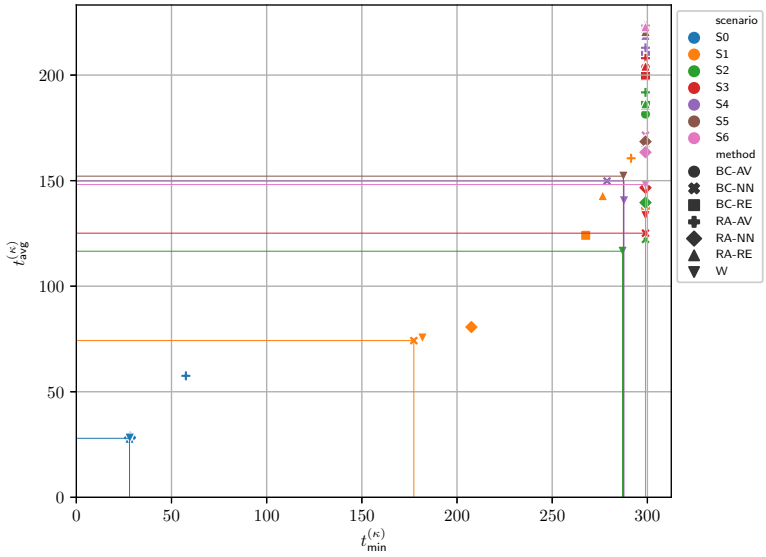


(b) Average vs Minimum Time to achieve $\kappa \geq 3$.

Figure 10.5: Average time vs minimum time to cover all stationary targets (i.e., not moving) with 1 (left) or κ agents (right). Results where agents are tasked to cover targets with $\kappa \geq 3$.



(c) Average vs Minimum Time to Coverage with at least one agent when tasked with $\kappa \geq 5$.



(d) Average vs Minimum Time to achieve $\kappa \geq 5$.

Figure 10.5: Average time vs minimum time to cover all stationary targets (i.e., not moving) with 1 (left) or κ agents (right). Results where agents are tasked to cover targets with $\kappa \geq 5$.

With the increase of number of targets in each scenario, the average and minimum times to coverage also increase. For each scenario $S1$–$S6$, there is a difference on average between the different methods. Mostly, the proposed method, indicated with the 'W' in the legend, is either the best on average or at least on the Pareto frontier, for scenarios $S3$ in Figure 10.5b; $S2$ and $S5$ in Figure 10.5c; $S2$, $S5$, and $S6$ in Figure 10.5d; and for scenarios $S2$ and $S3$ in Figure 10.5a; $S2$ in Figure 10.5b; $S4$ and $S6$ in Figure 10.5c; and $S4$ in Figure 10.5d, respectively. Similar performance is displayed by the *BC-NN* method, which is the best performing method among the ones considered in this study.agentagentagent

When only one target is involved, the average minimum time to coverage is lowest. In all cases, the agents will move in Z and eventually find and cover the targets. However, when increasing the number of targets ($S1$–$S6$), it can happen that agents gather on the first targets found, leaving remaining targets undiscovered for the rest of the simulation. As such, all metrics are affected, and the $t_{j,e}^{(\kappa)}$ is saturated to the duration of the simulation $T_{\text{sim}} - 1$ [2] for the targets that were not discovered. In Figure 10.5, the points are accumulated at the $t_{\min}^{(\kappa)} - 1$, which means that in those scenarios there were undiscovered targets for the whole duration of the simulation.

10.4.2 Results for Dynamic Targets

In our second set of experiments, targets move within the map Z by randomly changing direction, with velocity $v_{t,\max} = 1.5\ m$ per time-step. In both cases, agents move with a higher velocity $v_{a,\max} = 2\ m$ per time-step. Nevertheless, we still use the same sets of scenarios. As for the performance, for a single experiment $e = 1, \ldots, N$, we consider the average time for which a target j is covered with at least κ agents over the simulation, $\tau_{j,e}^{(\kappa)}$, and the average amount of agents that cover the target j over the simulation $\alpha_{j,e}$. Based on these two quantities we compute the following metrics: (i) the average time for which at least κ agents cover the targets, $\tau_{\text{avg}}^{(\kappa)}$, and (ii) the average amount of agents that cover the targets α_{avg}. These quantities are computed as

$$\tau_{\text{avg}}^{(\kappa)} = \frac{1}{N} \sum_{e} \frac{1}{|O|} \sum_{j} \tau_{j,e}^{(\kappa)} \tag{10.16}$$

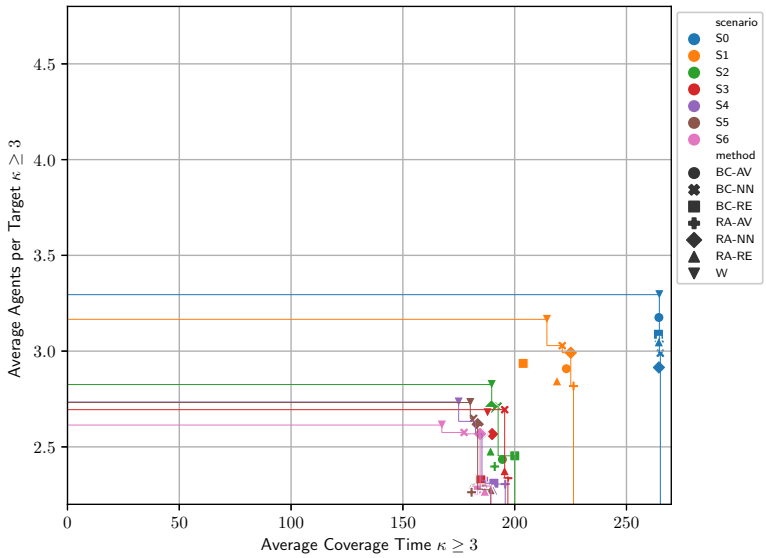
[2] In the final time-step the multi-agent system shuts down, hence this time-step is not considered when dealing with the results.

$$\alpha_{\text{avg}} = \frac{1}{N} \sum_{e} \frac{1}{|O|} \sum_{j} \alpha_{j,e} \tag{10.17}$$

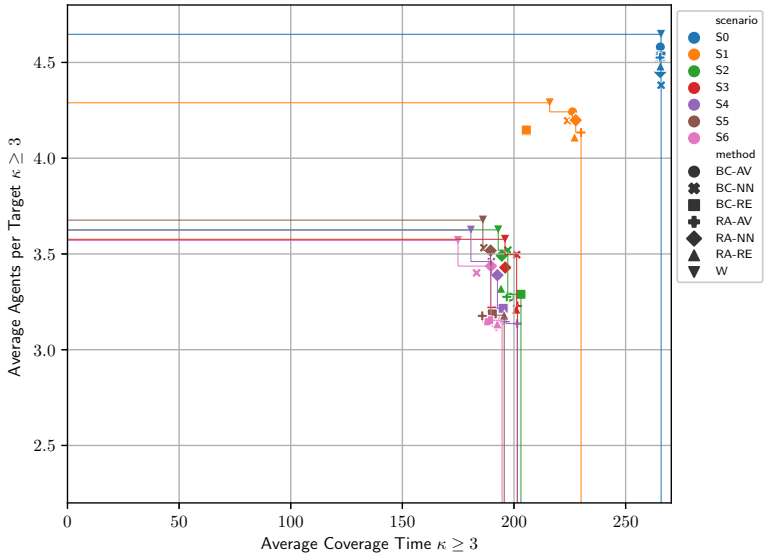
While in our first set of experiments, featuring a set of static targets, agents will cover a target for the whole duration of the simulation once they joined a coalition, in the dynamic case, such assumption cannot be made, because agents can lose targets as all objects are moving in Z. Furthermore, these metrics are calculated twice for *active* and *passive coverage*, i.e., by (i) considering the targets that agents are actively following by adjusting their own motion and being part of a coalition, and (ii) considering targets that are not being actively followed, but are within the visibility range of agents, without necessarily being part of a coalition.

Results are shown in Figure 10.6, where the average time of coverage is given along the x-axis, and the average number of agents is given on the y-axis. It is possible to observe that for $\kappa \geq 3$ the method with the willingness is overall on the Pareto frontier, with an exception for scenario $S3$, shown in Figure 10.6a. Regarding $\kappa \geq 5$, the method with the willingness, indicated with W in the legends of Figure 10.6, is on the Pareto frontier for scenarios $S0$–$S4$, and the best on average for $S5$–$S6$, Figure 10.6c. The same is observed for passive following in Figure 10.6d. Furthermore, our approach tends toward maximizing the number of agents covering a target, thus it lies on the left side of the Pareto frontier. Similarly to the results in the static case, the performance of the *BC-NN* strategy is comparable to the method with the willingness. We can observe that for both $\kappa \geq 3$ and $\kappa \geq 5$ the average coverage time is highest when the number of targets is lower, in $S0$ and $S1$, falls for $S2$–6 when the number of targets to be covered increases. We speculate that an increase in the number of targets, whilst the size of the area is unchanged, might increase the average coverage time as agents can join multiple coalitions. However, this remains subject to further research. The impact of the chosen values for κ can be observed as well in Figure 10.6, by inspecting the average coverage times, which are lower for $\kappa \geq 5$ than $\kappa \geq 3$. Taking into account what is being covered passively increases the average number of agents that cover a target.

Note that, the averages are taken over all time-steps of the simulation including the time to discover the objects in the first place. As such the lack of coverage before the discovery naturally penalizes the shown results.

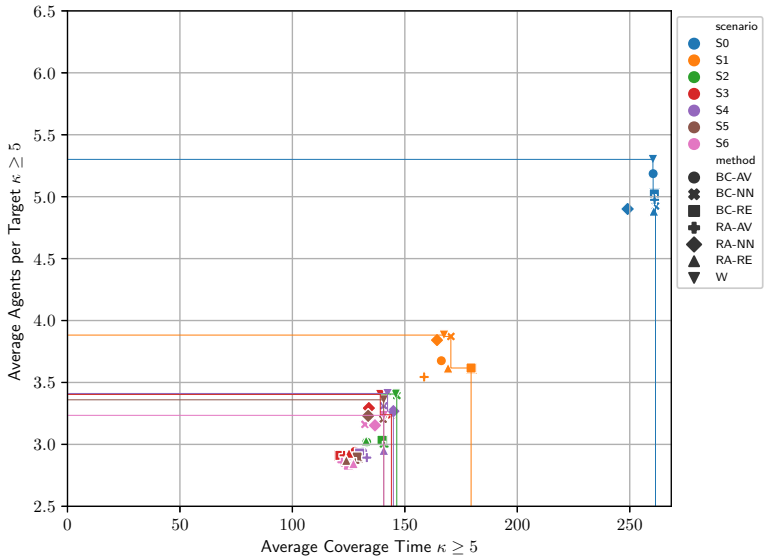


(a) Active Coverage $\kappa \geq 3$

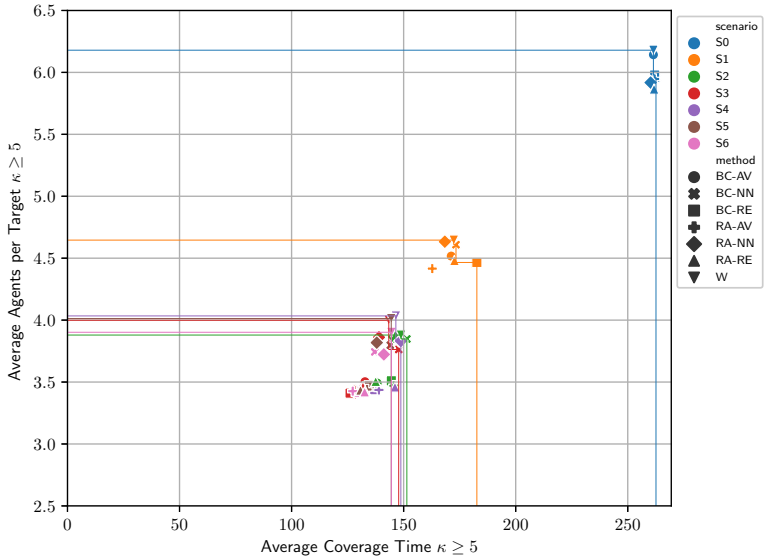


(b) Passive Coverage $\kappa \geq 3$

Figure 10.6: Average number of agents covering targets vs average coverage time of all targets of the entire duration of the simulation. We show both active and passive coverage. The results show the case where agents are tasked to cover targets with $\kappa \geq 3$.



(c) Active Coverage $\kappa \geq 5$



(d) Passive Coverage $\kappa \geq 5$

Figure 10.6: Average number of agents covering targets vs average coverage time of all targets of the entire duration of the simulation. We show both active and passive coverage. The results show the case where agents are tasked to cover targets with $\kappa \geq 5$.

In our current approach, the agents are not aiming to exceed the desired coverage. Nevertheless, this can happen due to race-conditions in the coalition formation process. Furthermore, this can also take place when an agent detects that a target is moving away. In this case it will try to find another agent that can join the coalition. At the same time, the target is not dropped by the former agent until it actually goes out of its visibility range while the new agent already joined the coalition and might have the target within its visible range.

10.5 Generalization of the Approach

In this paper, a collaborative approach based on the willingness to interact has been tailored to solve the κ-coverage problem for a multi-robot system. The described framework, composed of the agent behaviour, willingness to interact, and interaction protocol can be applied in other problems as well. Regarding the agent behaviour, the state machine presented in Section 10.3.2 can be generalized by considering the following abstract states: *idle*, *interact*, *idle & execute*, and *interact & execute* adapted from [13]. The latter can be specialized depending on the behaviours that the robots should have for solving different problems, e.g., moving by randomly changing direction and inspecting the space for new targets can be used to instantiate the *idle* state into the *inspect* state as done in this paper for solving the κ-coverage problem. Whereas the *execute* state can be instantiated into either the *inspect & follow* or *evaluate & follow*, by adding the target following behaviour to the agents.

The willingness to interact formalism can be easily adopted to account for additional relevant factors in a given application domain. The framework allows for the factors to be grouped into two categories, necessary and optional, as well as giving a specific weight to each factor. In this paper we have considered the battery level and the number of targets an agent is already following, which correspond to the necessary and optional factors respectively. Weights are determined in a simple way, i.e., if no necessary factor is under the the minimum threshold, then factors are weighted the same, otherwise the necessary factors will override the optional ones, thus determining the final value of the willingness.

Finally, the interaction protocol is independent of the application and problem to be solved, apart for the κ parameter which can be adjusted depending on the size of the coalitions that the robots should be able to form, and the triggers that agents use to initiate the interaction. In the current application domain agents are tasked with discovering and tracking targets in their environment.

Therefore, the triggers for executing the interaction protocol are application dependent such as (i) spotting a new target in the visibility range, (ii) detecting that a target is moving away and might soon be outside of the visibility range, and (iii) extending an existing coalition in order to reach κ-coverage. The fourth trigger captures the moment when an agent decides that it needs to ask for help, which is based on the willingness to interact. This trigger is not application dependent.

10.6 Conclusion and Future Work

This paper presented a novel, distributed, agent-centric coalition formation approach, based on the willingness to interact for adaptive cooperative behavior. We showed that we can use this novel approach to solve the $\kappa-$coverage problem for a set of targets. The performance of this approach is measured along two different sets of metrics for two different cases, (i) with static targets, and (ii) with mobile targets, and compared with six methods previously proposed in the literature. In the former case, the average time to get one target covered with κ agents, and the average minimum time to $\kappa-$cover all objects are considered. In the latter case, the average coverage time and average number of agents per target are considered. Results show that our approach either performs comparably good in the case of static targets with respect to the *BC-NN* method (the best performing among the ones considered in the paper), and that it performs better than the other methods in terms of achieving a higher level of coverage when it comes to moving targets.

There are three main lines of inquiry for future work. First, it is of interest to compare further the performance of our approach with those methods that reached similar performance like the *BC-NN* method. Such investigation can include the exploration of other experimental settings that might better highlight possible trade-offs for the utilization of the *BC-NN* method or the one based on the willingness proposed in this paper. Furthermore, issues related to how the studied models scale up in terms of, e.g., bandwidth capacity and latency, can also be considered in the analysis. Second, security aspects can be introduced, by considering the trustworthiness of agents. Such information can be included in the calculation of the willingness to interact, in order to facilitate the cooperation between agents that are more trustworthy, e.g., open systems where new agents may be introduced or removed, similarly to recent approaches [17, 18]. Third, some assumptions made in this paper can be relaxed, e.g., targets can appear and disappear at random times, or leave the area

defined by the map, in order to adapt the current approach for solving a more general κ-coverage problem.

Acknowledgements

This work was supported by the DPAC research profile funded by KKS (20150022), the FIESTA project funded by KKS, and the UNICORN project funded by VINNOVA.

Bibliography

[1] S. García, C. Menghi, P. Pelliccione, T. Berger, and R. Wohlrab. An architecture for decentralized, collaborative, and autonomous robots. In *Proc. of the Int. Conf. on Software Architecture*, pages 75–7509, 2018.

[2] Wolfram Burgard, Mark Moors, and Frank Schneider. Collaborative exploration of unknown environments with teams of mobile robots. In Michael Beetz, Joachim Hertzberg, Malik Ghallab, and Martha E. Pollack, editors, *Advances in Plan-Based Control of Robotic Agents*, pages 52–70, 2002.

[3] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1):165 – 200, 1998.

[4] Faisal Qureshi and Demetri Terzopoulos. Distributed coalition formation in visual sensor networks: A virtual vision approach. In James Aspnes, Christian Scheideler, Anish Arora, and Samuel Madden, editors, *Proc. of the Int. Conf. on Distributed Computing in Sensor Systems*, pages 1–20, 2007.

[5] Dayong Ye, Minjie Zhang, and D. Sutanto. Self-adaptation-based dynamic coalition formation in a distributed agent network: A mechanism and a brief survey. *IEEE Trans. on Parallel & Distributed Systems*, 24(5):1042–1051, 2013.

[6] G Theraulaz, E Bonabeau, and J-L Deneubourg. Response threshold reinforcements and division of labour in insect societies. *Proc. of the Royal Society B: Biological Sciences*, 265(1393):327–332, 1998.

[7] L. Esterle. Goal-aware team affiliation in collectives of autonomous robots. In *Proc. of the Int. Conf. on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 90–99, 2018.

[8] Lukas Esterle and Peter R. Lewis. Online multi-object k-coverage with mobile smart cameras. In *Proc. of the Int. Conf. on Distributed Smart Cameras*, pages 1–6. ACM, 2017.

[9] Lukas Esterle and Peter R. Lewis. Distributed autonomy and trade-offs in online multiobject k-coverage. *Computational Intelligence*, 2019.

[10] L. E. Parker and B. A. Emmons. Cooperative multi-robot observation of multiple moving targets. In *Proc. of the Int. Conf. on Robotics and Automation*, volume 3, pages 2082–2089 vol.3, 1997.

[11] Cyril Robin and Simon Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, 40(4):729–760, Apr 2016.

[12] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: An open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[13] Mirgita Frasheri, Baran Cürüklü, Mikael Ekström, and Alessandro Vittorio Papadopoulos. Adaptive autonomy in a search and rescue scenario. In *Proc. of the Int. Conf. on Self-Adaptive and Self-Organizing Systems*, pages 150–155, 2018.

[14] Roberto Casadei, Danilo Pianini, Mirko Viroli, and Antonio Natali. Self-organising coordination regions: A pattern for edge computing. In Hanne Riis Nielson and Emilio Tuosto, editors, *Coordination Models and Languages*, pages 182–199, Cham, 2019. Springer International Publishing.

[15] Seok-Hyoung Lee and Hoon Choi. The fast bully algorithm: For electing a coordinator process in distributed systems. In *Revised Papers from the International Conference on Information Networking, Wireless Communications Technologies and Network Applications-Part II*, ICOIN '02, pages 609–622, Berlin, Heidelberg, 2002. Springer-Verlag.

[16] A. Hellmund, S. Wirges, Ö. Ş. Taş, C. Bandera, and N. O. Salscheider. Robot operating system: A modular software framework for automated driving. In *Proc. of the Int. Conf. on Intelligent Transportation Systems*, pages 1564–1570, 2016.

[17] Eduardo Castelló Ferrer. The blockchain: A new framework for robotic swarm systems. In Kohei Arai, Rahul Bhatia, and Supriya Kapoor, editors, *Proceedings of the Future Technologies Conference (FTC) 2018*, pages 1037–1058, Cham, 2019. Springer International Publishing.

[18] Davide Calvaresi, Alevtina Dubovitskaya, Jean Paul Calbimonte, Kuldar Taveter, and Michael Schumacher. Multi-agent systems and blockchain: Results from a systematic literature review. In Yves Demazeau, Bo An, Javier Bajo, and Antonio Fernández-Caballero, editors, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection*, pages 110–126, Cham, 2018. Springer International Publishing.

Chapter 11

Adaptive Autonomy in Wireless Sensor Networks

Authors: Mirgita Frasheri, José Cano-García, Eva González-Parada, Baran Çürüklü, Mikael Ekström, Alessandro V. Papadopoulos, and Cristina Urdiales
Venue: 19^{th} International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'20)

Abstract

Moving nodes in a Mobile Wireless Sensor Network (MWSN) typically have two maintenance objectives: (i) extend the coverage of the network as long as possible to a target area, and (ii) extend the longevity of the network as much as possible. As nodes move and also route traffic in the network, their battery levels deplete differently for each node. Dead nodes lead to loss of connectivity and even to disengaging full parts of the network. Several reactive and rule-based approaches have been proposed to solve this issue by adapting redeployment to depleted nodes. However, in large networks a cooperative approach may increase performance by taking the evolution of node battery and traffic into account. In this paper, we present a hybrid agent-based architecture that addresses the problem of depleting nodes during the maintenance phase of a MWSN. Agents, each assigned to a node, collaborate and adapt their behaviour to their battery levels. The collaborative behavior is modeled through the willingness to interact abstraction, which defines when agents ask and give help to one another. Thus, depleting nodes may ask to be replaced by healthier counterparts and move to areas with less traffic or to a collection point. At the lower level, negotiations trigger a reactive navigation behaviour based on Social Potential Fields (SPF). It is shown that the proposed method improves coverage and extends network longevity in an environment without obstacles as compared to SPF alone.

11.1 Introduction

Wireless sensor networks (WSNs) consist of a number of interconnected spatially distributed nodes equipped with sensors to capture information from the environment. WSNs are fit for different scenarios, ranging from emergency deployment – where the network needs to be established as fast as possible – to long term monitoring — where the network must last as long as possible, e.g., sensors for irrigation or vibration sensors for seismic areas. In all cases, the network needs to provide an appropriate coverage of the target area. WSNs may be homogeneous, when all nodes have the same role, or heterogeneous. In both cases, WSNs typically include one or more sink nodes, Access Points (APs), to redirect gathered data to external networks. The locations of the APs are typically predefined and stationary, because they are usually connected to a main power supply to route all the network traffic to an external network. However, most nodes in WSNs are typically battery-operated. Battery life depends on connectivity and throughput, which in turn primarily depend the on deployment method [1]. Unfortunately, WSN deployment cannot be optimized in many scenarios and it becomes quite complex for large networks [2]. The locations of the nodes in the network could be computed with optimal approaches. However once the nodes begin to fail, those configurations are no longer optimal. Thus, optimal approaches are not suitable for WSN, since their nodes are prone to failure [3].

Mobile WSN (MWSN) try to solve this problem by providing nodes some degree of mobility, so that they can deploy themselves (self-deployment) or, at least, adjust their locations (self-healing) when other nodes start to fail, in order to cover the gaps that appear in the network [4]. Although there are other options, Multiple Robot Systems (MRS) have been often used in combination with MWSN to transport nodes when needed [3, 5, 6].

As in traditional autonomous robot navigation problems, there are deliberative and reactive approaches for MWSN navigation. Deliberative approaches are meant to optimize efficiency [3, 7–11], however, they require a reliable model of the environment – including network configuration, environment layout, traffic, etc. – and they are computationally expensive. Reactive approaches rely on local factors instead [12–16], e.g., setting a local behavior for each node or a set of local rules that the nodes must follow. The combination of all local behaviors provide an emergent global behaviour with reduced computational cost, where results are not optimized and local minima may appear. In order to avoid the drawbacks mentioned above, in the robotics domain it is common to combine both deliberative and reactive strategies into a hybrid approach.

A reactive approach to MWSN deployment and self-healing was proposed in [17, 18]. The proposed approaches are a variation of the Social Potential Fields (SPF) [19], originally proposed for navigation in a robot swarm. The underlying idea is that each robot is affected by a number of forces, namely: (i) a repulsion force depending on the Received Signal Strength (RSS) that leads to network dispersion; (ii) an attraction force depending on how many nodes each one is linked to that preserves connectivity; and (iii) a repulsion force depending on nearby obstacles that avoids collisions. Nodes in the MWSN move until the sum of these forces is equal to zero, i.e., the network is balanced. Then, they stay at their locations until a number of nodes run out of battery and the network needs to be balanced again. This algorithm successfully extended the network life preserving coverage as much as possible and outperformed rule-based systems like the Backbone algorithm [20]. However, given its reactive nature, it had a number of problems because no high level strategy was used to determine specific node locations. In this work, we propose to use an agent architecture to add a cooperative layer to the reactive system.

Agents have already been used in MRS, specifically in robot teams and swarms [21–23]. Coalition formation algorithms designed for multi-agent systems have been adapted for the multi-robot domain in order to solve tasks such as box-pushing, cleaning, and sentry duty [23, 24]. Cognitive agents have been integrated into robotic swarm architectures with the purpose of guiding the behaviour of the group without losing the advantages of the swarm approach [21]. Others have applied distributed scheduling mechanisms in robot teams operating in a hospital environment, mapping an agent to a robot, thus allowing each robot to compute its own schedule in a distributed way [22].

Another relevant aspect relates to the autonomy of agents involved in the decision-making. Autonomy can be defined based on the dependencies between agents, i.e., when an agent a needs the intervention of another agent b in order to complete its goal G, then a depends on b for G, and is not autonomous with respect to b in this context [25]. Furthermore, mechanisms with which agents change their levels of autonomy based on the circumstances, have also been investigated. They range from adjustable autonomy, where there is a human in the loop, an operator, which makes the decision on whether to change autonomy levels of the agents, to adaptive autonomy, where agents themselves decide their own level of autonomy [26].

In this work we present a hybrid control architecture for a swarm of robot MWSN nodes where the reactive layer is controlled by a SPF and the cooperative layer relies on an agent architecture that allows agents to adapt their autonomy during their operation. This agent approach is based on the willing-

ness to interact abstraction, which determines when agents ask and give help to one another [27]. As in most works involving a large number of robotic entities [3,6], the evaluation of our hypothesis – that the hybrid agent approach yields better performance than SPF alone – relies on simulations. However, the working parameters for the presented algorithm have been set using a reduced number of real robots. Results show that the proposed method increases the coverage in the network, as well as the network longevity as compared to SPF alone. Furthermore, the achieved improvement is consistent for the whole operation of the network.

The rest of this paper is organized as follows. Section 11.2 describes in detail the problem to be solved. Section 11.3 presents the previously proposed low level reactive navigation algorithm. Section 11.4 describes the hybrid agent approach and the willingness to interact abstraction that affects how agents collaborate with one another. Section 11.5 describes the design of the experiment, while Section 11.6 presents our results. Concluding remarks and an outline of directions for future work are given in Section 11.7.

11.2 Problem Formulation

Let's assume an area Z with known dimensions $\langle d_w, d_h \rangle$, and n agents[1] $a_i \in \{a_1, \ldots, a_n\}$ allowed to move in Z. Each agent is characterized by its battery level b_i, and it is able to connect to others within its communication range r_C. An agent's battery is impacted by: (i) its motion in Z; and (ii) the amount of traffic it routes. Therefore, agents will have different life-spans, depending on how much distance they have covered, and how much traffic they have routed. The amount of traffic depends on the topology of the network and on the routing algorithm, thus, it is very hard to predict a priori in MWSN. Similarly, the robot motion in a configuration with a large number of agents is also hard to predict. Instead, these parameters are measured by each node through its life.

Agents have two goals, (i) to provide as much coverage for the network as possible, and (ii) to extend the longevity of the network as much as possible. Coverage depends on the network topology, so it is important to distribute robots adequately. The longevity of the network depends on the lifespans of the robots in the MWSN. Therefore, the second goal is addressed by both minimizing the travelled distance of the agents, which aims at potentially extending their lifespans, and by distributing the traffic load uniformly among nodes.

[1] In this paper the terms agent, robot, and WSN node are used interchangeably, as we have one agent per robot.

11.3 SPF

As commented, the reactive system consists of a cooperative agent based layer and a reactive low level algorithm. Specifically, the reactive layer uses SPF algorithm proposed previously [17] for deployment and self-healing. The original SPF was proposed for autonomous navigation in swarm robots by Reif and Wang [19]. SPF defines attraction and repulsion forces to set goals and/or constraints. In our system, our goals are: i) to spread the network to achieve the maximum possible coverage with living nodes; ii) to preserve connectivity among living nodes; and iii) to avoid collisions when the robots move. Hence, three forces are defined.

- Obstacle repulsion force(s) $f_{r1}(r_{i,j})$ repel the robot from other robots or physical obstacles in its vicinity to prevent collisions. This force is only significant in the vicinity of physical objects.
- Deployment repulsion force(s) $f_{r2}(r_{i,j})$ moves robots away from each other to spread them and increase coverage.
- Cohesion force (connectivity attraction) $f_c(r_{i,j})$ increases with $r_{i,j}$ to avoid loss of communication among nodes.

$r_{i,j}$ being the distance between robots i and j. In order to estimate relative distances between robots and with respect to the coverage area boundaries, we need a node localization technique. The locations of all elements under simulation is always known. In real tests, locations could be estimated by combining robot odometry and RSS-based trilateration [28, 29].

The combination of all three forces for each node in the network returns an emergent motion vector. A node stops moving when that vector is under a threshold f_u. When all nodes stop, the network is balanced. Afterwards, if any force changes for a node, e.g., nearby nodes die or they have to move significantly for one reason or another, that node moves again. If it affects other nodes in the region, the network will need to be balanced again. This process is repeated in time until the performance of the network is deemed insufficient after enough nodes have died.

It can be observed that there is no high level strategy in the robot motion at this point. Nodes simply respond to the defined forces and no plan is made to increase the network life by considering the specifics of each node. The main novelty of the present work is the addition of an agent layer over the SPF. We assign an agent to each robot in the network, which is aware of the different

parameters that the robot uses to calculate SPF forces, in addition to its battery level and routed traffic. Agents may decide that some robots need to switch location with others when their battery level is not fit to route the traffic in the area. Thus, by purposefully redirecting nodes when necessary to adapt to the network traffic specifics, we increase its average life. The following sections describe this proposal in detail.

11.4 Agent Approach

This section describes how a cooperative agent approach based on the willingness to interact is combined with the presented reactive SPF method, in order to address the problem described in Section 11.2 for maximising the coverage and extending the longevity of a wireless sensor network. A detailed account is given on how agents compute their willingness to interact, and on the negotiation protocol used.

11.4.1 Agent Behaviour in the MWSN

The operation of agents in a wireless sensor network follows three phases, initialization, deployment, and maintenance. During the first phase agents are started and their state variables are initialized. Afterwards, the deployment phase follows using SPF, in which agents move away from the initial locations in order to extend coverage of the target area as much as possible, while keeping the connectivity to the AP. When the deployment is complete and the network is stabilized, agents continue draining their batteries due to traffic routing. As commented, not all agents have the same battery level after deployment, as they have not travelled the same distance. Furthermore, depending on their location and the routing algorithm, some may route far more traffic than others. Hence, some agents may be depleted before others. If an agent is depleted, it may stay on location as a new obstacle or travel back to a battery charging station, leaving a ”hole” in the network. When a number of agents have died, the SPF needs to be run again to re-stabilize the network. Alternatively, rather than waiting for nodes to be depleted, the proposed approach gets nodes with higher battery levels to replace the ones that are about to be depleted. Thus, allowing for a more graceful degradation of the coverage, and extension of the life of the network.

Let’s assume a critical battery level b_{l0} defined as the amount of battery that an agent needs to go to a collection point, e.g., an AP. This value could

be different for every agent, as it depends on where in the network an agent is positioned, or we could have a safe threshold that guarantees that a node can reach the collection point from any location in the coverage area for simplicity. Furthermore, let's assume two battery levels b_{lh} and b_{l1} defined as $b_{l0} \leq b_{lh} \leq 10\% \cdot b_{l0} + b_{l0}$, and $b_{l0} \leq b_{l1} \leq 30\% \cdot b_{l0} + b_{l0}$, respectively. The thresholds are selected heuristically and are used to indicate how close a robot's battery is to its critical level.

The first agent to reach the b_{l0} triggers the first negotiation round in the network. Then, all agents with battery level below $b_{l1}^{(t)}$ start sending help requests to the network to ask agents with battery above $b_{l1}^{(t)}$ to replace them. The urgency of a request sent by an agent in need, as well as the disposition to help by other agents is captured in the willingness to interact, as described in the next subsection. The agents with battery above $b_{l1}^{(t)}$ are the ones that will respond to these help requests and move to new locations if so decided.

After a negotiation round is complete, all agents with battery level below $b_{lh}^{(t)}$ move towards their collection point. These agents are too depleted to be useful anymore. Note that we do not remove only the one node that reached $b_{l0}^{(t)}$, but a percentile of nodes that will reach it in the near future. Since most nodes will probably move during the balancing stage, nodes with the lowest battery would probably reach the removal threshold shortly after. Hence, setting two thresholds avoids running SPF each time a single node needs to leave. After the assignment to locations is complete, negotiation-winning agents move towards their target positions. Nodes with battery between $[b_{l0}^{(t)}, b_{l1}^{(t)}]$ do not move at all. During this motion stage, only obstacle repulsion forces are active to avoid collisions. Finally, after all moving agents have reached their target locations, the SPF is run again in order to balance the network.

11.4.2 Willingness to Interact

The willingness to interact defines a general disposition of an agent to interact with other agents, by either asking or giving help [27]. The willingness at time t, $w^{(t)} \in [-1,1]$, depends on the internal state of an agent, and is not related to any particular task or help request that may have been received from others. In this paper, the calculation of the willingness to interact is influenced only by the current battery level b_c. Note, further, that the previously proposed equation [27] has been adapted in order to include WSN related parameters.

The willingness to interact at a time t is given by,

$$w^{(t)} = \begin{cases} -1, & \text{if } b_c^{(t)} \leq b_{l0}^{(t)}, \\ -\frac{b_c^{(t)} - b_{l0}^{(t)}}{b_c^{(t)}}, & \text{if } b_{l0}^{(t)} < b_c^{(t)} \leq b_{l1}^{(t)}, \\ \frac{b_c^{(t)} - b_{l1}^{(t)}}{b_c^{(t)}}, & \text{if } b_{l1}^{(t)} < b_c^{(t)}. \end{cases} \tag{11.1}$$

This equation states that when an agent's battery is below $b(t)_{l1}$, the corresponding willingness will be negative, and the agent is in a state where it is ready to ask for help. Furthermore, the closer $b_c^{(t)}$ is to $b_{l0}^{(t)}$, the more negative the willingness will be. If the value for the willingness goes below the threshold $w_H = -\frac{b_{lh}^{(t)} - b_{l0}^{(t)}}{b_c^{(t)}}$ corresponding to $b_{lh}^{(t)}$, the agent might soon reach the critical battery level. When at least one agent's willingness value becomes -1, i.e., at least one has reached the critical battery level, then agents below w_H start asking for help and moving toward the collection point, whereas agents with $w_H \leq w \leq 0$ stay where they are and start asking for help. If $b_{l1}^{(t)} \leq b_c^{(t)}$ then an agent's willingness is positive, therefore the agent can respond to help requests from others in the WSN.

Once a request for help is initiated, agents with positive willingness will reason to meet this request. Furthermore, all requests for agents below w_H are considered, whereas for the others only those requests where the willingness is not more than 20% over w_H are considered. This threshold is also set heuristically, and is used to indicate which agents are closest to the hysteresis level. The willingness specifies the overall disposition of an agent to interact based on its own state. Nevertheless, when an agent gets a request for help, the willingness needs to be refined to reflect the trade-off between staying in the current position with a particular traffic load, and moving to a new position with another traffic load. This adjustment is done by incorporating in the willingness the utility of an agent for moving to a new position with a known traffic load (Equation 11.2).

$$W^{(t)} = w^{(t)} + u^{(t)}, \tag{11.2}$$

where $u^{(t)}$ is the utility for moving to a new position, calculated as,

$$u^{(t)} = \begin{cases} 1 - \frac{b_m^{(t)} + b_n^{(t)}}{b_c^{(t)}} + \frac{\frac{b_c^{(t)}}{b_A^{(t)}} - \frac{b_c^{(t)}}{b_B^{(t)}}}{\frac{b_c^{(t)}}{b_A^{(t)}} + \frac{b_c^{(t)}}{b_B^{(t)}}}, & \text{if } 1 - \frac{b_m^{(t)} + b_n^{(t)}}{b_c^{(t)}} \geq 0.3, \\ -w^{(t)}, & \text{otherwise,} \end{cases} \tag{11.3}$$

where $b_m^{(t)}$ is the battery required to move to the new position, $b_n^{(t)}$ is the battery level to go from the new position to the collection point, $b_A^{(t)}$ is the battery spent on routing per iteration in the current position, and $b_B^{(t)}$ is the battery that would be spent for routing in the new position. This means that, if an agent has at least 30% of battery on top of what is needed to go to a new position, and the collection point thereafter, then it proceeds with reasoning on whether to help. Otherwise, the utility is set to $-w^{(t)}$ and, given Equation 11.2, the willingness of the agent will be 0, thus it will not take part in the negotiation. When an agent has enough battery for useful motion, then the impact of the traffic load is also considered. If the new position requires more battery, then the utility is decreased, otherwise it is increased. In brief, agents with more battery will favor more traffic intense locations and vice-versa.

In this paper, the willingness to interact is only dependent on the battery level, however, in other applications there might be other state variables to consider, e.g., number of tasks needed to be achieved concurrently. The same consideration holds for the calculation of the utility.

11.4.3 Negotiation Protocol

The negotiation protocol is triggered when at least one agent in the system reaches its critical battery level b_{l0}, or when the *AP* identifies a dead node in the network that did not ask for help in time. In the former case, agents with negative willingness will send a help request to all alive nodes in the network, collect the responses, assign the agent with the highest positive willingness, and notify the assigned agent by sending a packet in the network. In the latter case, the *AP* will send a help request to all alive agents on behalf of the dead node. Moreover, the AP handles only one dead dead per negotiation round. Afterwards, from the responses, the agent with the highest positive willingness is assigned to the location of the dead robot, and notified. Agents with positive willingness will process the requests and send their answers back to the requesting agents. A request will be considered if it comes from those agents below w_H, as well as those agents with willingness not more than 20% on top of w_H.

All additional packets generated due to the negotiation between agents, and how they affect the battery drainage, are computed. For the calculation of the number of packets that flow through the network and are routed by the nodes, we assume that the network uses a limited flooding strategy known as geographic routing, according to which each packet sent by an origin towards

a destination is re-transmitted by other nodes but only if they are nearer to the destination. This type of strategy is widely used in WSN when the network deployment is not planned and signaling traffic needs to be very limited to extend the network life as much as possible [3, 30]. More details are provided in Section 11.5.2.

11.5 Experiment Design

This section describes the hypothesis investigated in this paper, the metrics used for the evaluation of our approach and its comparison with SPF, the communication model, and the simulation setup used for generating the data used in the analysis.

11.5.1 Hypothesis and Evaluation Metrics

The hypothesis investigated in this paper is formulated as follows:

Hypothesis 1. *The hybrid approach that combines agent collaboration with a reactive layer for adaptation, yields better results with respect to network coverage and longevity of the network in the maintenance phase of a MWSN, as compared to a solely reactive approach.*

In order to evaluate the hypothesis, the following metrics are defined: blanket coverage, which addresses the coverage concern, and energy efficiency and consumption which indirectly address the longevity of the network. Blanket coverage refers to any point of the area of interest covered by at least one node [31]. Assume a node i that covers an area A_i. Then the coverage for n nodes over the whole area A is calculated as:

$$C = \frac{\bigcup_{i=1,\ldots,n} A_i}{A}. \tag{11.4}$$

Equation 11.4 is transformed into a probabilistic model [32] as follows:

$$C = \sum_{i=1}^{m} \frac{p_i}{m}, \tag{11.5}$$

where p_i is the probability of detecting an event in cell i, in a probabilistic grid with m cells. The probability is given by:

$$p_i = 1 - \overline{p}_j = 1 - \prod_{j=1}^{n} (1 - p_{ij}), \tag{11.6}$$

where p_{ij} is the probability that node j detects an event at location i.

Aspects that relate to energy are captured on one hand by the uniformity of the network U for n nodes (Equation 11.7).

$$U = \frac{1}{n}\sum_{s=1}^{n} U_s, \tag{11.7}$$

$$U_s = \sqrt{\frac{1}{K_s}\sum_{j=1}^{K_s}(r_{s,j} - \bar{r}_s)^2}, \tag{11.8}$$

where $K_{s,j}$ represents the number of nodes close to node s, $r_{s,j}$ the distance between nodes s and j, and $\bar{r}_s$ the average distance between s and its closest neighbours.

On the other hand, such aspects are also reflected in the average power consumed by the nodes to send a message $\overline{P}$ (Equation 11.9).

$$\overline{P} = \frac{1}{n}\sum_{j=1}^{n} \overline{P}_j, \tag{11.9}$$

$$\overline{P}_j = \frac{1}{n}\sum_{j=1}^{n-1} P_{s,j}, \tag{11.10}$$

$$P_{s,j} = P_{s1} + \cdots + P_{sk}, \tag{11.11}$$

where $\overline{P}_j$ is the average power for node j to send a packet to the network, and $P_{s,j}$ is the power consumed to send a message from node s to j, over k hops needed for the packet to reach the destination.

Finally, the overhead produced by the packets generated due to the negotiation between agents is estimated and compared to the total number of packets routed in the network.

11.5.2 Simulation Setup

The hypothesis has been evaluated in simulation[2], by comparing the performance of SPF alone [18], with the proposed hybrid approach combining both SPF and agent collaboration. The environment used for the evaluation of the SPF approach, proposed previously [17], has been extended to support agent negotiation, and relies on the tools provided by the Player/Stage simulator [33].

[2]The code for running the simulations is publicly available at https://bitbucket.org/gitagent/gitagent_wsn/src/master/

The simulated area has a size of $150x150\text{m}^2$, scale 1 : 10, and is populated by $n = 100$ robots. Each robot is initiated with a battery level $b_0 = 3000\text{mAh}$. Only the sink is assumed to be connected to the power supply, and will not deplete. We have considered two main causes of battery drainage: the routing of packets through the network and the movement of the robots.

Assuming that the robots move at a constant speed, the charge drained from the battery (in $\text{mA} \cdot \text{h}$) when a robot moves 1m can estimated as:

$$b_{1m} = \frac{m_C}{v \cdot 3600} = 1.11\text{mA} \cdot \text{h/m}, \tag{11.12}$$

where $m_C = 200\text{mA}$ is the electric current that flows through the motor while the robot is moving, and $v = 0.05\text{m/s}$ is the robot speed, in accordance with the consumption model of the Hexbug robot toys derived in previous work [18].

The other cause of battery drainage is the network operation, which is heavily influenced by the nature of the data traffic and also by the behaviour of the communication protocols, and in particular by the routing strategy and the medium access control (MAC) mechanism. A typical low-power and short-range communication transceiver requires a supply current of 10-20mA during its transmission and reception operations [34]. Therefore, each time a node sends a packet some battery charge is consumed, but also the battery is drained when reception is enabled (either if data is actually being received or not). The MAC strategy of the link layer determines how much time the transceiver spends in each possible state (transmission, reception or sleep) during its operation. In order to save battery, a duty-cycling MAC strategy is typically used to allow the transceiver to remain in a low-power state most of the time [35] while maintaining the network operational. Thus, our simulation model considers two main sources of battery consumption due to network operations: a constant background consumption due to the duty-cycling operation, which is equal for every node in the network, and the consumption caused by packet transmissions, which depends on how many packets each node is sending and/or routing.

Assuming that the MAC parameters are properly tuned, the average current drained during duty-cycling operation can be as low as 1% of the current required when the transceiver is in the receiving state [36], so in our simulation model we have considered that $0.15\text{mA} \cdot \text{h}$ are drained from the battery each hour of network operation. On the other hand, duty-cycling strategies require each packet to be repeatedly transmitted over a period of time to guarantee it is finally received, which increases the battery waste of sending a packet. In

our simulation we assume a wake-up frequency of 20 Hz for the duty-cycling protocol, so each packet should be transmitted repeatedly during a 50ms interval. The supply current of the transceiver during the transmission operation depends slightly on transmission power, which in turn determines transmission range. In accordance with the datasheet of a commercial transceiver [34], our model considers a 12.5mA supply current during transmission for a coverage range of 25m. Regarding all this, the charge drained from the battery each time a packet is sent and/or routed by a node is set to $1.74 \cdot 10^{-4}\text{mA} \cdot \text{h}$. To calculate the amount of packets routed by each node, both the data traffic sent by the sensor nodes to the AP and the packets sent during the agent negotiation rounds have to be regarded. A simple model has been considered for the data traffic sent by the nodes to the AP during network operation: each node periodically tries to send one packet to the AP (at a rate of one packet every 10 seconds), which is routed through the network by intermediate nodes. Also, during the agent negotiations, some packets have to be exchanged between nodes requesting help, nodes willing to help, and the AP, and these packets are also routed by intermediate nodes in order to reach their destinations. As mentioned in Section 11.4.3, a geographic routing mechanism is adopted to compute the number of packets each nodes routes. Geographic routing operates over multi-hop mesh topologies and its main advantages when compared to other routing strategies for WSN are that it provides a reasonable scalability, adapts quickly to network topological changes, and is stateless and, therefore, does not require additional signaling for routing [30,37]. Our simulation model implements one of the simplest approaches to geographic routing, the closest neighbor routing: a given node i broadcasts a packet that is received by all of its neighbor nodes within its transmission range. The receiving neighbor nodes broadcast the packet to their own coverage zone, but only if they are closer to the destination node than the one from which they received the message. A node never re-transmits the same packet more than once. This algorithm generates a partial flooding, but is very simple and reasonably energy-efficient.

In regards to robot motion, the parameters of the three different forces applied to each robot except the AP, have been obtained heuristically, and depend on the modeled physical robots. The obstacle repulsion force is calculated as:

$$f_{r1}(r_{i,j}) = -\frac{0.001}{(r_{i,j})^8}. \tag{11.13}$$

The deployment repulsion force is calculated as:

$$f_{r2}(r_{i,j}) = -\frac{20}{(r_{i,j})^7}. \tag{11.14}$$

The cohesion force is calculated as:

$$f_c(r_{i,j}) = -\frac{\alpha \cdot n_{fail}}{n \cdot n_{nearAP}}, \tag{11.15}$$

where n_{fail} is the number of depleted nodes, and n_{nearAP} is the number of nodes directly connected to the AP (i.e., located within the coverage area of the AP).

The simulation will stop when more than 70% of the robots have depleted their battery, i.e., their current battery level is lower than 5% of the initial battery level. It is assumed that the locations of all the robots, as well as the boundaries of the area are known. In a physical environment the robots would rely on triangulation using their own RF signals for localization. Ideally, the sink node will be static as well, so its position will be also known. Usually, at least two more beacons (with known positions) are used in the deployment area so the other nodes can triangulate themselves. Of course, triangulation errors may be up to a few meters and the error will also propagate for nodes that triangulate themselves using information from mobile nodes instead of beacons. Nevertheless, the error will not accumulate over time, as the positions are recalculated from the RSS at every new triangulation. More importantly, these errors are not critical for the application – at most, robots will end a few meters away from the destination or re-transmit some unnecessary packets. If localization is critical, more beacons could be added where necessary. Finally, there are no obstacles in the environment, but dead robots with depleted batteries are treated as obstacles.

11.6 Results

The comparison between the combined agent approach and the SPF was done by considering two values for the α parameter in the calculation of the cohesion force $f_c(r_{i,j})$ active in the network, $\alpha \in \{5, 20\}$. Thus, there are two configurations for each approach, namely (i) $SPF1$ with the base-line value for the cohesion force defined as in previous work ($\alpha = 20$) [17, 18], (ii) $SPF2$ with reduced cohesion force ($\alpha = 5$), (iii) $AA1$ with the base-line cohesion force, and (iv) $AA2$ with reduced cohesion force. Note that, for $AA1$ and $AA2$, the battery waste for the packets generated due to the negotiation is taken into account in the battery consumption of each robot.

Simulations for each of the four cases were repeated 30 times, with the corresponding means and standard deviations (see Table 11.1) taken over the values of each metric at the step where less than 50% of the nodes are alive, and

Table 11.1: Statistics for the metrics over 30 runs.

	Statistics for 50% nodes dead							
	cov.		dist.(m)		uniform.		time(days)	
	avg	std	avg	std	avg	std	avg	std
$SPF1$	0.45	0.008	84.44	1.32	0.53	0.0056	65.84	1.85
$SPF2$	0.02	0.001	49.81	0.56	0.43	0.0110	106.83	5.96
$AA1$	0.48	0.012	94.44	4.55	0.51	0.0074	63.89	2.04
$AA2$	0.67	0.021	80.13	3.44	0.71	0.0152	75.77	3.37
	Statistics at simulation end							
	cov.		dist.(m)		uniform.		time(days)	
	avg	std	avg	std	avg	std	avg	std
$SPF1$	0.30	0.0088	102.23	2.05	0.45	0.0091	110.2	4.6
$SPF2$	0.30	0.2435	62.76	2.12	0.54	0.1118	209.0	10.9
$AA1$	0.28	0.0050	144.95	8.62	0.39	0.0113	108.7	2.7
$AA2$	0.41	0.0097	109.73	7.53	0.55	0.0298	139.5	5.5

at the end of the simulation. Results in Figures 11.1-11.7 consist of the curves for each simulation run, as well as the average over 30 runs, for every method. Visually, methods can be distinguished by the color hue, with bolder lines for the averages. A direct average of the curves is not possible because each run yields different length of the simulation. The average is therefore computed considering up to the shortest simulation for the considered method.

The averaged values (Table 11.1) show the state of the network under the different methods for two different time points as aforementioned. It can be seen that when the 50% mark for the dead nodes is reached, $AA2$ achieves better coverage and uniformity with respect to the rest, whereas with respect to the distance walked it performs better than $AA1$. Moreover, $SPF2$ is the last to reach the 50% mark, followed by $AA2$, with $SPF1$ and $AA1$ being the fastest to reach such level of dead nodes. At the end of the simulation, $AA2$ maintains higher levels of the coverage, with the rest of the methods showing slightly worse performance. Regarding uniformity, at the end of the simulation $AA2$ and $SPF2$ are rather comparable, outperforming the other two. In terms of walked distance and longevity of the network, the results are similar to the results for the 50% mark. During the operation of the network, as time passes and nodes become depleted, the $AA2$ method consistently achieves higher coverage than the others (Figures 11.1-11.2) [3]. Note that an area A_i is considered

[3]Note that the sampling frequencies for the four methods is different. This is because the SPF-

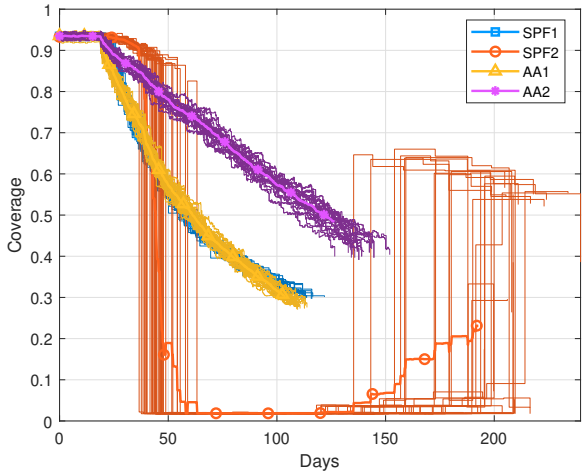


Figure 11.1: Coverage with respect to time

covered if the covering node is connected to the AP, either directly, or through several hops. The nodes close to the AP route higher amounts of traffic, as compared to the nodes in the outskirts of the network. Thus, they become depleted faster. When the cohesion force is reduced for $SPF2$, and the central nodes start to die, the network does not shrink to keep the connectivity to the AP. Instead a gap is created in the network (see Figure 11.3). The resulting coverage goes close to 0. Nevertheless, it can happen that the nodes manage to come closer to the AP and reconnect at the end of the simulation, thus improving on the coverage as well as the other metrics. In the case of $AA2$, when central nodes start to die, they are replaced by nodes at the outskirts, as such the nodes remain connected to the AP. The nodes at the outskirts are the first to move because their battery levels are highest, which is due to the low traffic load. It is possible to observe that the $AA2$ method yields consistently better coverage, throughout the whole operation time of the network, oppositely to $SPF2$ which can by chance recuperate close to the end of the simulation.

The network, in terms of alive nodes (Figure 11.4), lasts the longest for the $SPF2$ method. However, in this case alive nodes become disconnected

based methods are only rerun when the nodes breakdown and the network should balance. In between, the alive nodes are stationary. However, for the agent-based methods, the negotiations start taking place before any nodes breakdown, i.e., the time between failures of nodes is simulated as well.

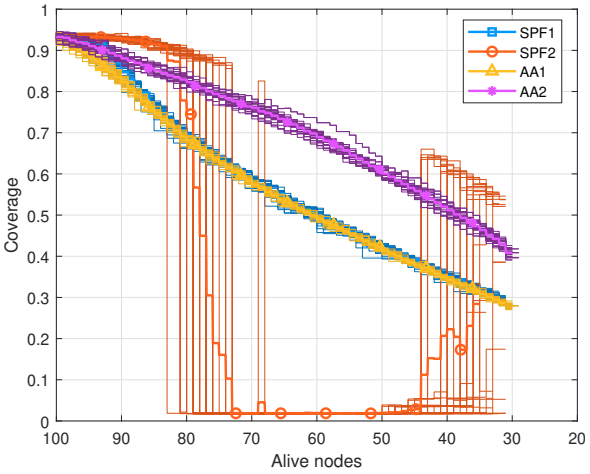
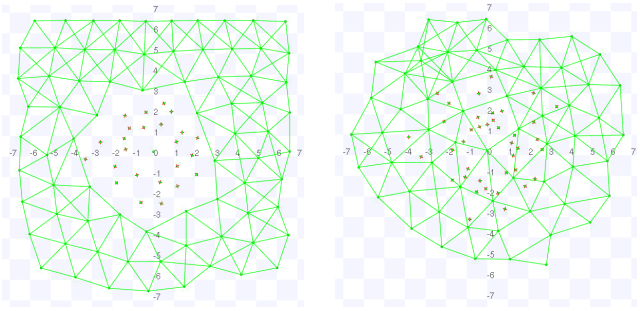


Figure 11.2: Coverage with respect to the number of alive nodes

from the AP. The $SPF1$ and $AA1$ methods are comparable in terms of depleted nodes over time, nevertheless with $SPF1$ nodes last longer. The $AA2$ method has a consistently lower rate of depleted nodes over time as compared to $SPF1$ and $AA1$, resulting in highest network longevity. The life of the network is extended with circa 29 days on average as compared to $SPF1$. With respect to the distance traveled by nodes (Figure 11.5), $SPF1$ and $AA2$ are comparable, whereas $AA1$ has the highest distance traveled. The $SPF2$ method has the lowest traveled distance, due to the reduced cohesion force, the nodes will not move towards the AP to keep the connectivity.

The $AA1$ and $AA2$ methods generate additional packets in the network due to the negotiation between agents overtime. The averages for the total amount of negotiation packets over the total amount of routed packets in the network are 2.1×10^{-4} and 1.8×10^{-4}, for $AA1$ and $AA2$ respectively (Figure 11.6). Therefore, the battery consumption due to this overhead is negligible.

The $AA2$ method outperforms the other methods in our tests with respect to uniformity (Figure 11.7). Whereas, $SPF1$ and $AA1$ are comparable until the 65^{th} day, approximately. Afterwards, as the nodes keep dying, uniformity is preserved better with $SPF1$. Similarly to the coverage metric, for some experiments $SPF2$ regains connectivity close to the end of the simulation, thus improving on the uniformity as well.



a) SPF2 b) AA2

Figure 11.3: Robot layout on the simulated map for an intermediate step of the simulation.

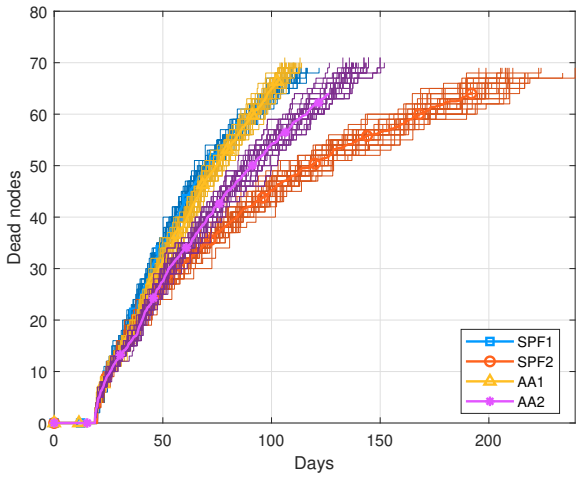


Figure 11.4: Depleted nodes with respect to time

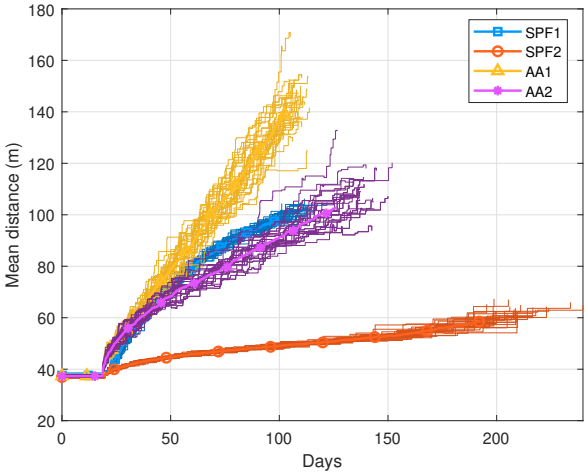


Figure 11.5: Distance traversed with respect to time. Note that the y-axis starts from 20 meters

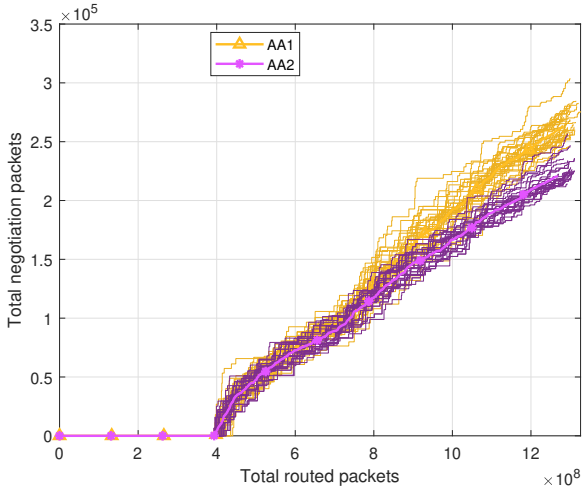


Figure 11.6: Number of negotiation packets vs the total number of routed packets.

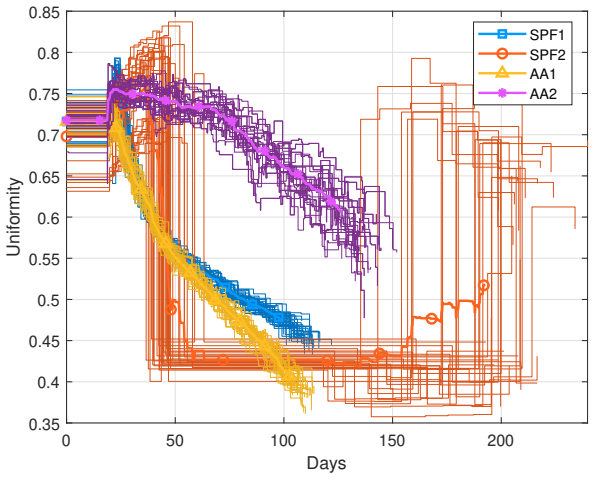


Figure 11.7: Uniformity in the network with respect to time. Note that the y-axis starts from 0.35 units.

11.7 Conclusion

A hybrid agent approach, which combines a reactive layer with explicit collaboration between agents for the self-healing phase in mobile wireless sensor network has been presented. The reactive layer is based on a SPF algorithm. The agent collaboration part is modeled through the willingness to interact abstraction, which defines when agents ask and give help to each other based on their battery level. It is shown that the hybrid agent approach improves the coverage and longevity of the network, as compared to the social potential fields algorithm. Furthermore, the proposed approach does not require a high cohesion force to keep the nodes connected to the AP. Instead, this comes as a result of how agents negotiate with one another. After a negotiation, nodes at the outskirts of the network with low traffic load, take the place of central nodes, which become depleted at a faster pace.

There are four lines of inquiry for future work. Firstly, the comparison between the hybrid agent approach and the social potential fields algorithm can be extended to account for environments with obstacles. Secondly, machine learning techniques can be used to adjust the forces for each node in the network, depending on individual battery levels, and traffic load. Thirdly, other delegation strategies could be investigated, and compared to the current work

where agents drop their current positions when assigned to new ones. Furthermore, the reassignment of agents could be negotiated before the critical battery level is reached, in order to send depleting agents in locations with less traffic. Lastly, it is of interest to evaluate the proposed approach for hierarchical networks with several levels of the nodes.

11.8 Acknowledgements

The work in this paper was funded by the VINNOVA project UNICORN, the DPAC research profile (KKS funding 20150022), by the FIESTA KKS project, by the Swedish Foundation for Strategic Research under the project "Future factories in the cloud (FiC)" with grant number GMT14-0032, and by the Spanish project RTI2018-096701-B-C21 and the PY18-1652 Andalusian regional project.

Bibliography

[1] Joshua Robinson, Eugene Ng, and Joshua Robinson. A performance study of deployment factors in wireless mesh networks. In *in IEEE Infocom, 2007*, pages 2054–2062, 2007.

[2] Daniel-Ioan Curiac. Towards wireless sensor, actuator and robot networks: Conceptual framework, challenges and perspectives. *Journal of Network and Computer Applications*, 63:14–23, mar 2016.

[3] Vlajic N. and Moniz N. Self-healing wireless sensor networks: Results that may surprise. In *Proc. of the IEEE Global Telecommunications Conference Workshops. GLOBECOM Workshops*, pages 333–338, 2007.

[4] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Comput. Netw.*, 52(12):2292–2330, 2008.

[5] Nojeong Heo and P. K. Varshney. Energy-efficient deployment of intelligent mobile sensor networks. *Trans. Sys. Man Cyber. Part A*, 35(1):78–92, jan 2005.

[6] A. Howard, M. Mataric, and G. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks, 2002.

[7] Konstantinos P. Ferentinos and Theodore A. Tsiligiridis. Adaptive design optimization of wireless sensor networks using genetic algorithms. *Computer Networks*, 51(4):1031–1051, 2007.

[8] Nikitha Kukunuru, Babu Rao Thella, and Rajya Lakshmi Davuluri. Sensor deployment using particle swarm optimization. *International Journal of Engineering Science and Technology*, 2(1):5395–5401, 2010.

[9] Raghavendra V. Kulkarni and Ganesh Kumar Venayagamoorthy. Particle Swarm Optimization in Wireless-Sensor Networks: A Brief Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(2):262–267, mar 2011.

[10] Xiaoxiang Han. Mobile node deployment based on improved probability model and dynamic particle swarm algorithm. *Journal of Networks*, 9(1):131–137, 2014.

[11] Jin Yang, Fagui Liu, Jianneng Cao, and Liangming Wang. Discrete particle swarm optimization routing protocol for wireless sensor networks with multiple mobile sinks. *Sensors*, 16(7):1081, 2016.

[12] Novella Bartolini, Tiziana Calamoneri, Emanuele Guido Fusco, Annalisa Massini, and Simone Silvestri. Push & Pull: Autonomous deployment of mobile sensors for a complete coverage. *Wireless Networks*, 16(3):607–625, 2010.

[13] EA Jensen and ML Gini. Rolling Dispersion for Robot Teams. *IJCAI*, 2013.

[14] Jiming Chen, Shijian Li, and Youxian Sun. Novel deployment schemes for mobile sensor networks. *Sensors*, 7(11):2907–2919, 2007.

[15] Chun Zhang and Shumin Fei. Connectivity-Preserved and Force-Based Deployment Scheme for Mobile Sensor Network. *Wireless Personal Communications*, 77(1):463–475, nov 2013.

[16] Recep Özdağ and Ali Karcı. Probabilistic dynamic distribution of wireless sensor networks with improved distribution method based on electromagnetism-like algorithm. *Measurement*, 79:66–76, feb 2016.

[17] Cristina Urdiales, Francisco Aguilera, Eva González-Parada, Jose Cano-García, and Francisco Sandoval. Rule-based vs. behavior-based self-deployment for mobile wireless sensor networks. *Sensors*, 16(7):1047, 2016.

[18] Eva González-Parada, Jose Cano-García, Francisco Aguilera, Francisco Sandoval, and Cristina Urdiales. A social potential fields approach for self-deployment and self-healing in hierarchical mobile wireless sensor networks. *Sensors*, 17(1):120, 2017.

[19] John H. Reif and Hongyan Wang. Social potential fields: A distributed behavioral control for autonomous robots, 1999.

[20] S. Damer, L. Ludwig, M. Anderson LaPoint, M. Gini, N. Papanikolopoulos, and J. Budenske. Dispersion and exploration algorithms for robots in unknown environments. *Unmanned Systems Technology VIII, SPIE Digital Library*, 2006.

[21] F Aznar, Mireia Sempere, FJ Mora, Pilar Arques, JA Puchol, Mar Pujol, and Ramón Rizo. Agents for swarm robotics: Architecture and implementation. In *Highlights in Practical Applications of Agents and Multiagent Systems*, pages 117–124. Springer, 2011.

[22] Simon Thiel, Dagmar Häbe, and Micha Block. Co-operative robot teams in a hospital environment. In *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 2, pages 843–847. IEEE, 2009.

[23] Lovekesh Vig and Julie A Adams. Coalition formation: From software agents to robots. *Journal of Intelligent and Robotic Systems*, 50(1):85–118, 2007.

[24] Lovekesh Vig and Julie A Adams. Multi-robot coalition formation. *IEEE transactions on robotics*, 22(4):637–649, 2006.

[25] Cristiano Castelfranchi. Founding agent's 'autonomy' on dependence theory. In *Proceedings of the 14th European Conference on Artificial Intelligence*, pages 353–357. IOS Press, 2000.

[26] Benjamin Hardin and Michael A Goodrich. 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*, pages 165–172. ACM, 2009.

[27] Mirgita Frasheri, Baran Çürüklü, Mikael Ekström, and Alessandro Papadopoulos. Adaptive autonomy in a search and rescue scenario. In *12th IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, pages 150–155, September 2018.

[28] Shaifull Nizam Othman. Node positioning in zigbee network using trilateration method based on the received signal strength indicator (\(rssi\)). *European Journal of Scientific Research*, 46(1):048–061, 2010.

[29] K Aamodt. Cc2431 location engine. *Application Note AN042 (Rev. 1.0), SWRA095, Texas Instruments*, 2:2–4, 2006.

[30] Luis Javier Garcia Villalba, Ana Lucila Sandoval Orozco, Alicia Triviño Cabrera, and Claudia Jacy Barenco Abbas. Routing protocols in wireless sensor networks. *Sensors*, 9(11):8399, 2009.

[31] D. W. Gage. *Command control for many-robot systems*. Naval Command Control and Ocean Surveillance Center, RDT and DIV San Diego, CA, 1992.

[32] Amitabha Ghosh and Sajal K. Das. Review: Coverage and connectivity issues in wireless sensor networks: A survey. *Pervasive Mob. Comput.*, 4(3):303–334, 2008.

[33] Brian Gerkey, Richard T Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th international conference on advanced robotics*, volume 1, pages 317–323, 2003.

[34] Texas Instruments. *CC2500 Low-Cost Low-Power 2.4 GHz RF Transceiver*, 2016. Rev. C.

[35] Fayez Alfayez, Mohammad Hammoudeh, and Abdelrahman Abuarqoub. A survey on mac protocols for duty-cycled wireless sensor networks. *Procedia Computer Science*, 73:482 – 489, 2015.

[36] Adam Dunkels. *The contikimac radio duty cycling protocol*. Swedish Institute of Computer Science, 2011.

[37] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: A survey. *Comput. Netw. ISDN Syst.*, 47(4):445–487, mar 2005.