Non-Cooperative Games for Self-Interested Planning Agents

Jaume Magí Jordán Prunera

A thesis submitted for the degree of
Título de Doctor por la Universitat Politècnica de València

Supervisors: Prof. Eva Onaindía de la Rivaherrera
Dr. Mathijs M. de Weerdt
Dr. Alejandro Torreño Lerma

September 2017

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

A la meva família.

Acknowledgements. Agraïments.

Durant la realització d'aquesta tesi doctoral hi ha hagut moltes persones que, d'una manera o d'una altra, han ajudat a fer que aquest treball avançara. En aquestes línies vull agrair el seu suport.

En primer lloc, done les gràcies als meus tres supervisors, sense els quals tot açò no hagués sigut possible.

A Eva por darme la oportunidad de trabajar con ella, por transmitirme sus conocimientos y su espíritu trabajador, así como por las discusiones productivas que han ayudado a mejorar toda esta tesis.

To Mathijs for accepting me as a visiting scholar, and then for being my supervisor. Thank you for being a great researcher and supervisor, for teaching me so many things and sharing your time and knowledge to advise my work. Without your help, much of this work would not have been possible.

A mi amigo y compañero Alejandro, al que he podido incorporar en la última etapa de esta tesis para poder rematarla. Gracias por tu amistad, por las charlas y los buenos momentos, así como por tu apoyo y positividad en las situaciones más duras.

En segon lloc, m'agradaria donar les gràcies a Vicente Julián i a Stella, per introduir-me en el món de la investigació i ensenyar-me a donar les primeres passes. Sense ells tampoc hauria arribat fins ací.

Totes les hores passades al DSIC han sigut molt més agradables amb els companys del GTI-IA, i en especial, dels membres dels laboratoris 205 i 208. Gràcies a Elena, Juanmi, Joan, Víctor, Bexy, Sergio, María, Pablo, César, Jaime, Ángelo, Javi, Luís, Valentina, Martí, Natalia, Jose, Bogdan, Jesús, Debashis, Mohannad, Alexander, Diego, Carlos, Andrés, Vicent B., Aida, Fany, Sole, Laura,

Eliseo, Óscar, Miguel Ángel i Toni. Thanks to my Delft colleagues for making my research stay a good experience. Special thanks to Gleb, Simon, Erwin, Rens, Frits, Shruti, Peter, Matthijs S., and Cees.

Gràcies al GH team: Mario, Joan E., Flores, Joan P. i Miquel. Per riure tant, pels dinars, les discussions, els reptes, les festes, els consells, les confidències, l'esport, i totes les vivències passades i les que ens queden!

Als amics de sempre, Raul, Xavi, Martí, Maribel, Mayra, Núria, Laia, Jorge, Dani, Susana, Mari Carmen i Adrià (allà on estigues), perquè el temps simplement millora la nostra amistat. A Merche, per fer-me riure, pels bons moments i per tot el suport en la fase final d'aquesta tesi.

Als meus germans músics de Monográvido: Víctor, Dani i Pablo, per compartir la passió musical, per fer-ho sonar tot tan bé, per les tertúlies, les cerveses, i per ajudar-me a despertar a "Mi otro yo".

Als meus pares, per inculcar-me bons valors i motivar-me sempre en tot. A la meva teta Elena, i a Jordi, perquè créixer sense ells no hagués sigut el mateix i per sempre esperar-me amb pancartes. Als meus nebots Aleix i Ariadna, per contagiar-me la seua alegria, i per sempre voler jugar amb mi. Als meus *abuelos* i als avis, per transmetre'm el seu esperit lluitador. I en general a tota la meva família.

Gràcies a totes aquelles persones que han passat per la meua vida, que m'han ajudat i inspirat molt més del que podrien imaginar. I finalment, gràcies a la música, per connectar la vida, les emocions i els sentiments.

Abstract

Multi-Agent Planning (MAP) is a topic of growing interest that deals with the problem of automated planning in domains where multiple agents plan and act together in a shared environment. In most cases, agents in MAP are cooperative (altruistic) and work together towards a collaborative solution. However, when rational self-interested agents are involved in a MAP task, the ultimate objective is to find a joint plan that accomplishes the agents' local tasks while satisfying their private interests.

Among the MAP scenarios that involve self-interested agents, *non-cooperative MAP* refers to problems where non-strictly competitive agents feature common and conflicting interests. In this setting, conflicts arise when self-interested agents put their plans together and the resulting combination renders some of the plans non-executable, which implies a utility loss for the affected agents. Each participant wishes to execute its plan as it was conceived, but congestion issues and conflicts among the actions of the different plans compel agents to find a coordinated stable solution.

Non-cooperative MAP tasks are tackled through *non-cooperative games*, which aim at finding a stable (equilibrium) joint plan that ensures the agents' plans are executable (by addressing planning conflicts) while accounting for their

private interests as much as possible. Although this paradigm reflects many real-life problems, there is a lack of computational approaches to non-cooperative MAP in the literature.

This PhD thesis pursues the application of non-cooperative games to solve non-cooperative MAP tasks that feature rational self-interested agents. Each agent calculates a plan that attains its individual planning task, and subsequently, the participants try to execute their plans in a shared environment. We tackle non-cooperative MAP from a twofold perspective. On the one hand, we focus on agents' satisfaction by studying desirable properties of stable solutions, such as optimality and fairness. On the other hand, we look for a combination of MAP and game-theoretic techniques capable of efficiently computing stable joint plans while minimizing the computational complexity of this combined task. Additionally, we consider planning conflicts and congestion issues in the agents' utility functions, which results in a more realistic approach.

To the best of our knowledge, this PhD thesis opens up a new research line in non-cooperative MAP and establishes the basic principles to attain the problem of synthesizing stable joint plans for self-interested planning agents through the combination of game theory and automated planning.

Resumen

La Planificación Multi-Agente (PMA) es un tema de creciente interés que trata el problema de la planificación automática en dominios donde múltiples agentes planifican y actúan en un entorno compartido. En la mayoría de casos, los agentes en PMA son cooperativos (altruistas) y trabajan juntos para obtener una solución colaborativa. Sin embargo, cuando los agentes involucrados en una tarea de PMA son racionales y auto-interesados, el objetivo último es obtener un plan conjunto que resuelva las tareas locales de los agentes y satisfaga sus intereses privados.

De entre los distintos escenarios de PMA que involucran agentes auto-interesados, la *PMA no cooperativa* se centra en problemas que presentan un conjunto de agentes no estrictamente competitivos con intereses comunes y conflictivos. En este contexto, pueden surgir conflictos cuando los agentes ponen en común sus planes y la combinación resultante provoca que algunos de estos planes no sean ejecutables, lo que implica una pérdida de utilidad para los agentes afectados. Cada participante desea ejecutar su plan tal como fue concebido, pero las congestiones y conflictos que pueden surgir entre las acciones de los diferentes planes fuerzan a los agentes a obtener una solución estable y coordinada.

Las tareas de PMA no cooperativa se abordan a través de *juegos no cooperativos*, cuyo objetivo es hallar un plan conjunto estable (equilibrio) que asegure que los planes de los agentes sean ejecutables (resolviendo los conflictos de planificación) al tiempo que los agentes satisfacen sus intereses privados en la medida de lo posible. Aunque este paradigma refleja muchos problemas de la vida real, existen pocos enfoques computacionales para PMA no cooperativa en la literatura.

Esta tesis doctoral estudia el uso de juegos no cooperativos para resolver tareas de PMA no cooperativa con agentes racionales auto-interesados. Cada agente calcula un plan para su tarea de planificación y posteriormente, los participantes intentan ejecutar sus planes en un entorno compartido. Abordamos la PMA no cooperativa desde una doble perspectiva. Por una parte, nos centramos en la satisfacción de los agentes estudiando las propiedades deseables de soluciones estables, tales como la optimalidad y la justicia. Por otra parte, buscamos una combinación de PMA y técnicas de teoría de juegos capaz de calcular planes conjuntos estables de forma eficiente al tiempo que se minimiza la complejidad computacional de esta tarea combinada. Además, consideramos los conflictos de planificación y congestiones en las funciones de utilidad de los agentes, lo que resulta en un enfoque más realista.

Bajo nuestro punto de vista, esta tesis doctoral abre una nueva línea de investigación en PMA no cooperativa y establece los principios básicos para resolver el problema de la generación de planes conjuntos estables para agentes de planificación auto-interesados mediante la combinación de teoría de juegos y planificación automática.

Resum

La Planificació Multi-Agent (PMA) és un tema de creixent interès que tracta el problema de la planificació automàtica en dominis on múltiples agents planifiquen i actuen en un entorn compartit. En la majoria de casos, els agents en PMA són cooperatius (altruistes) i treballen junts per obtenir una solució col·laborativa. No obstant això, quan els agents involucrats en una tasca de PMA són racionals i auto-interessats, l'objectiu últim és obtenir un pla conjunt que resolgui les tasques locals dels agents i satisfaci els seus interessos privats.

D'entre els diferents escenaris de PMA que involucren agents auto-interessats, la *PMA no cooperativa* se centra en problemes que presenten un conjunt d'agents no estrictament competitius amb interessos comuns i conflictius. En aquest context, poden sorgir conflictes quan els agents posen en comú els seus plans i la combinació resultant provoca que alguns d'aquests plans no siguin executables, el que implica una pèrdua d'utilitat per als agents afectats. Cada participant vol executar el seu pla tal com va ser concebut, però les congestions i conflictes que poden sorgir entre les accions dels diferents plans forcen els agents a obtenir una solució estable i coordinada.

Les tasques de PMA no cooperativa s'aborden a través de *jocs no cooperatius*, en els quals l'objectiu és trobar un pla conjunt estable (equilibri) que asseguri que els plans dels agents siguin executables (resolent els conflictes de planificació) alhora que els agents satisfan els seus interessos privats en la mesura del possible. Encara que aquest paradigma reflecteix molts problemes de la vida real, hi ha pocs enfocaments computacionals per PMA no cooperativa en la literatura.

Aquesta tesi doctoral estudia l'ús de jocs no cooperatius per resoldre tasques de PMA no cooperativa amb agents racionals auto-interessats. Cada agent calcula un pla per a la seva tasca de planificació i posteriorment, els participants intenten executar els seus plans en un entorn compartit. Abordem la PMA no cooperativa des d'una doble perspectiva. D'una banda, ens centrem en la satisfacció dels agents estudiant les propietats desitjables de solucions estables, com ara la optimalitat i la justícia. D'altra banda, busquem una combinació de PMA i tècniques de teoria de jocs capaç de calcular plans conjunts estables de forma eficient alhora que es minimitza la complexitat computacional d'aquesta tasca combinada. A més, considerem els conflictes de planificació i congestions en les funcions d'utilitat dels agents, el que resulta en un enfocament més realista.

Des del nostre punt de vista, aquesta tesi doctoral obre una nova línia d'investigació en PMA no cooperativa i estableix els principis bàsics per resoldre el problema de la generació de plans conjunts estables per a agents de planificació auto-interessats mitjançant la combinació de teoria de jocs i planificació automàtica.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automated planning is the art of computing a course of action or *plan* which achieves the goals of a task from an initial state. Planning has been traditionally regarded as a centralized process in which a single entity builds a plan that satisfies the goals of the planning task (Ghallab, Nau, et al. 2004).

Multi-Agent Planning (MAP) is a research field which pursues the integration of planning capabilities in intelligent agents (Weerdt et al. 2009). MAP deals with the problem of automated planning in domains where multiple agents plan and act together in a shared environment (Nguyen et al. 2009). This research field is significant since it uses Artificial Intelligence (AI) techniques to compute a joint plan that achieves the agents' goals, which is used in different strata of society like industry, market management, space exploration, video games, e-science, or military applications (Clement 2005). The type of the participating agents in a MAP task defines the cooperative, collaborative or competitive nature of the MAP approaches. Agents can be classified as altruistic or non-strategic, in which case they do not have private interests and all

agents work together to solve a common task, and self-interested or strategic agents, where each agent is mostly concerned with the accomplishment of its individual planning task and it is usually motivated by the desire to maximize its own utility. A MAP approach capable of attaining for the agents' private interests will imply a significant impact in the society since it would benefit all involved entities, especially in industry contexts where the monetary profit is the main objective.

The work in (desJardins, Durfee, et al. 1999) examines *distributed planning* for non-strategic agents from two different perspectives; one approach regards a MAP task as the process of formulating or executing a plan among a number of participants; the second approach puts the focus on coordinating and scheduling the actions of multiple agents in a shared environment. The first approach has evolved to the so-called *cooperative MAP*, where multiple agents plan cooperatively to achieve a common goal, irrespective of how the goals, the knowledge and the agents' abilities are distributed in the application domain (Durfee 2001). On the other hand, the objective of the second line is controlling the execution of multiple agents aimed at solving a common task in order to ensure that the agents' local goals are met in a global context. This research line, commonly known as *decentralized planning*, is mostly concerned with problems such as task decomposition, resource allocation or reducing communication overhead in distributed coordination (Lesser et al. 2004).

When rational and self-interested agents are involved in a MAP task, the objective is to find a joint plan that accomplishes the agents' local tasks while retaining their private interests. In this scenario, common and conflicting interests are involved. That is, because conflict of interest arises when formulating or executing their plans in a shared environment, every party usually looks for an agreement which is as favorable as possible (cooperation). This type of pro-

blems are better addressed with *game theory*, the study of mathematical models of conflict and cooperation between intelligent rational and self-interested agents (Von Neumann and Morgenstern 2007). The mathematical analysis of game theory arises naturally when observing and analyzing a conflict from a rational point of view. A *game* is thus a conflicting situation in which opposing interests of individuals or institutions prevail. In this context, the decision or *strategy* of one party influences the decision that the others will make, and hence, the result of the conflict is determined from the decisions made (or strategies applied) by all the parties. One objective pursued by game theory is to predict a stable solution (equilibrium), a solution in which no agent will be better off by changing its strategy unilaterally (Shoham et al. 2009, Chapters 3 and 12). In a MAP task with self-interested agents, game theory is necessary to guarantee stable solutions that naturally emerge from the rational behavior of agents, while still keeping their private interests instead of complying with the authority of a centralized or external entity.

Among the different types of games, cooperative and non-cooperative game theory apply to strategic management behavior, a key feature in self-interested planning agents. In Cooperative Game Theory (CGT) agents compete but also cooperate to improve utility by joining coalitions. Thus, the aim of CGT is to predict which coalitions agents will form, the joint actions that groups take and the resulting collective payoffs (Brafman et al. 2009; Dunne et al. 2010).

Non-Cooperative Game Theory (NCGT), in contrast, tries to predict the agents' individual strategies and payoffs in order to find stable solutions without contracting each other's behavior since cooperation is self-enforcing. In a NCGT scenario applied to MAP, agents do not create value by joining coalitions or contracting others; each agent solves its planning task without communicating with the other partners and regardless of how the others solve their tasks. When rational planning agents are strictly competitive, the MAP task is formu-

lated as an scenario where agents directly compete against each other. The goal of an agent in this setting is to defeat or hurt the other agents and hence, the gains of an agent are precisely the losses of the other agents, and vice versa. For instance, in a domain where several companies compete to attract customers, the larger the number of customers for a company, the larger the benefits for this company, and the lower the benefits for the rest of the companies. Fully competitive approaches to MAP are known as *adversarial MAP* and they are generally based on zero-sum games (Bercher et al. 2008; Sailer et al. 2007).

There exist, however, other type of games inside NCGT, known as non-strictly competitive games, which feature competitive and cooperative elements. This type of game is regarded as a conflicting-interest coordination game, where a conflict is a loss of utility to an agent when it applies its strategy alongside the other agents' strategies. From a game-theoretical perspective, a non-strictly competitive game is addressed as *general-sum game* or non-zero sum game (Shoham et al. 2009, Chapter 3) (Osborne et al. 1994), where the gains of an agent are not the other agents' losses and so there can be win-win situations. Hence, agents seek to satisfy their private interests but not at the cost of hurting others as in zero-sum games. In a MAP task, conflicts arise when self-interested agents put their plans all together and the combination of plans render some plans non-executable. Every partner wishes to execute its independently computed plan but interactions among the actions of the plans prevent some agents from executing its plan as it was originally devised. This compels agents to find a coordinated stable solution.

This PhD thesis pursues the application of non-strictly competitive games to solve MAP tasks with rational and self-interested agents, a problem that has been largely neglected in the literature. We will refer to this type of problems as **non-cooperative MAP** tasks. Each agent calculates a plan for its planning

task and subsequently all agents try to execute their plans in a shared environment. Planning conflicts arise due to synchronization problems between actions (simultaneous use of the same variable (Blum et al. 1997)) or because the execution of some actions invalidates the conditions for the execution of subsequent actions of other agents' plans (potential clobbering actions between the plans of different agents (Cox et al. 2009)). Therefore, agents opt for coordinating their plans in a game context while resolving conflicts, which typically results in a loss of utility with respect to its standalone plan. Otherwise, if all agents impose the execution of their plans, an unresolved conflict will lead to non-executable plans and so agents will fail achieving the goals of their planning tasks. In these problems, it is necessary to find a stable (equilibrium) joint plan that ensures the agents' plans are executable (by avoiding planning conflicts) while accounting for their private interests as much as possible. This paradigm reflects real-life problems like traffic flow regulation, where agents have complementary interests because they all wish a smooth driving avoiding collisions, but also conflicting interests due to the use of the same roads, what may cause a decrease of the utility of their driving plan or strategy.

While there has been some research in the field of non-cooperative MAP, this is not a well-studied problem. The work in (Larbi et al. 2007) interleaves planning and execution by defining a game where agents select the actions to execute at each time step. The focus of this approach is to determine the set of goals that are achievable by each partner but agents are not able to make a long-term reasoning to achieve a given set of goals. The best-response planning approach proposed in (Jonsson et al. 2011) solves congestion games (Rosenthal 1973), a particular type of games where the payoff of each player depends on the resources it chooses and the number of players choosing the same resource. Conflicts managed in this approach are all due to the simulta-

neous use of resources but other planning conflicts are neglected. All in all, we can affirm there is a lack of computational approaches for non-cooperative MAP to address the problem of building a stable joint plan.

This PhD thesis opens up a new research line in non-cooperative MAP and establishes the basic principles to attain the problem of synthesizing stable joint plans for self-interested planning agents through the combination of game theory and automated planning. The theoretical models presented in this PhD thesis aim to address real-world problems that require coordination of a set of self-interested planning agents. We propose two different models to tackle this task from a twofold perspective. The first model assumes agents have a set of precomputed plans and the objective is that each agent selects and schedules one plan from the set such that the combination of all the schedules yields a stable solution. In this first model, we focus on agents' satisfaction by studying different measures of optimality and fairness. The second model considers agents with an unlimited set of plans, i.e., agents have planning capabilities to generate plans from their search space, so they can build their plans on the basis of the others' plans. This is not an infinite set since the plans are limited by the costs and the search space of the agents. We also improve efficiency by using a game-theoretical technique to iteratively compute a stable solution. We study convergence to stable solutions, and we show good complexity results under some assumptions. The theoretical models are implemented and tested through several experiments in order to empirically prove our hypotheses. The empirical evaluation of the models is a significant contribution since it is not common in game-theoretic approaches.

1.1 Objectives

The main objective of this PhD thesis is to propose two models to solve non-cooperative planning problems among self-interested agents that interact in a shared environment. In the following, we present the objectives of this work as well as the associated tasks tackled throughout this research and the resulting contributions:

1. Review of the related work. We revise the literature regarding the main topics of this research.

   (a) Review of the basic concepts and related work of classical planning and MAP with non-strategic agents.

   (b) Review and analysis of the existing approaches that combine game theory and planning.

2. Design of a game-theoretic model for agents with a limited set of pre-computed plans:

   (a) Definition of a theoretical model to obtain equilibrium solutions.

   (b) Analysis of stable solution concepts, Pareto optimality, and fairness to enhance the satisfaction of the agents in a game-theoretic setting.

   (c) Development of algorithms to solve the non-cooperative MAP task.

   (d) Analysis of computational complexity.

3. Design of a non-cooperative MAP model for agents with an unlimited set of plans:

(a) Definition of the theoretical model that allows agents to build their plans on the basis of the others' plans.

(b) Design of a mechanism for conflict handling that enhances conflict solving.

(c) Extension of the model to deal with congestion games by designing a mechanism to avoid congestion with delays.

(d) Analysis of the complexity of the task to find out under which conditions it can be efficiently solved with the proposed model.

4. Implementation and validation of the two game-theoretic MAP models.

(a) Design and implementation of algorithms to schedule the actions of the agents' plans as a non-cooperative game (first model).

(b) Adaptation of an existing multi-agent planner to implement our own non-cooperative multi-agent planner (second model).

5. Empirical evaluation of the two models to validate the hypotheses of this PhD thesis in various non-cooperative MAP tasks.

(a) Experiments with planning domains adapted to a non-cooperative setting that features planning conflicts among agents.

(b) Comparison of the approach presented in (Jonsson et al. 2011) and analysis of upsides and downsides of our proposal.

(c) Experiments with domains that feature congestion.

(d) Specification and experimentation with a case study based on a real-world problem that features electric and autonomous vehicles. This

MAP task includes all the elements that self-interested agents must tackle, i.e., conflicts, congestion, and individual cost functions.

(e) Analysis of the individual agents' strategic behavior.

1.2 Document Structure

The remainder of this document is organized as follows:

- Chapter 2 analyzes the related work in non-cooperative MAP. First, we review classical planning and MAP with non-strategic agents. Then, we revise existing non-cooperative MAP approaches which present common characteristics with the work of this PhD thesis. This chapter covers objective 1.

- Chapter 3 presents FENOCOP (Fair Equilibria in NOn-COoperative Planning), the initial non-cooperative MAP model designed in the context of this PhD thesis. FENOCOP is a game-theoretic approach that includes two different games: the General Game, which considers the agents' plans as strategies and obtains equilibrium solutions, and the Scheduling Game, where agents reason about how to schedule these plans in order to attain executable joint solutions that are Pareto optimal and fair equilibria. This model covers the objective 2 of the present work.

- Chapter 4 presents an empirical evaluation of FENOCOP, the model introduced in Chapter 3. The two games of FENOCOP, the General Game and the Scheduling Game, are fully implemented, which accounts for objective 4.a. We perform an experimental evaluation of the performance and quality of the FENOCOP algorithms by means of different non-cooperative MAP tasks, thus objective 5.a is totally covered.

- Chapter 5 presents BRPS (Better-Response Planning Strategy), an approach for non-cooperative MAP where a set of self-interested planning agents reach an equilibrium through better-response dynamics. BRPS considers congestion and conflicts as part of the agents' utility. We analyze the complexity of BRPS, studying under which conditions our approach behaves as a potential game. We deeply analyze convergence to different equilibria, reporting positive complexity results under some assumptions. This chapter attains the objective 3 of this PhD thesis.

- Chapter 6 presents a comprehensive experimental evaluation of BRPS, the theoretical model introduced in Chapter 5. We compare our approach to BRP (Jonsson et al. 2011), a state-of-the-art best-response-based model that presents many similarities with our approach. The experiments include a congestion-based domain and several MAP domains adapted from the Competition of Distributed and Multi-Agent Planners (CoDMAP) (Komenda et al. 2016) benchmarks that feature planning conflicts. Additionally, we test a complex custom domain that includes both congestions and conflicts. This chapter covers objectives 4.b and 5.

- Chapter 7 presents the concluding remarks of this PhD thesis, the future research lines, and the author's related research activities and scientific publications.

Chapter 2

Related Work

In this chapter, we present the background more closely related to the research lines of this PhD thesis. Albeit there has been a rather intensive effort in investigation on Multi-Agent Planning (MAP), with some proposals focused on game-theoretic planning, there are hardly proposals that tackle non-strictly competitive MAP tasks. We will refer to this type of MAP tasks as *non-cooperative MAP* as indicated in the Introduction of this PhD thesis.

The assumptions adopted in MAP concerning the nature of the participating agents give rise to a great variety of MAP paradigms that range from altruistic planning, which assumes that agents are fully cooperative and do not have a strategic behavior or private interests, to *adversarial MAP*, where agents are self-interested and competitive. Figure 2.1 shows a hierarchy of MAP, where the left branch depicts approaches that feature altruistic agents and the right branch presents the research areas that deal with self-interested agents.

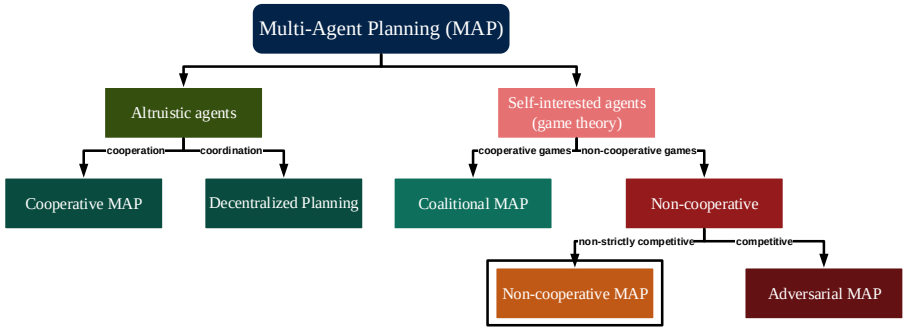


Figure 2.1: Multi-Agent Planning hierarchy

The mainstream of MAP approaches that involve agents with no private interests (left branch of Figure 2.1) is the distributed resolution of a common MAP task among multiple agents. Approaches that define non-strategic planning agents are also characterized by putting the emphasis on the cooperation of all the agents to solve a common task or on the coordination of the local plans of the agents. In *cooperative MAP*, agents typically combine their efforts to achieve a set of common goals. In *decentralized planning*, the objective is to coordinate the agents' plans in order to achieve their goals in a global context.

The right branch of Figure 2.1 includes the research lines that feature self-interested planning agents. We can identify two main approaches. On the one hand, addressing a MAP task as a cooperative game where agents form coalitions that help them achieve their goals is known as *coalitional MAP*. On the other hand, the application of non-cooperative games to MAP gives rise to two types of tasks: strictly competitive and non-strictly competitive. This PhD thesis is devoted to analyze non-cooperative MAP, the study of non-strictly competitive games to solve MAP tasks.

This chapter is structured as follows. Section 2.1 introduces the most important concepts of classical single-agent planning as well as some related work in cooperative MAP and decentralized planning, which are the most common MAP approaches that deal with non-strategic agents. In Section 2.2, we first describe the related work in coalitional MAP and adversarial MAP, and then we analyze the existing non-cooperative MAP approaches. Finally, we present the conclusions of this related work.

2.1 Planning with Non-strategic Agents

In this section, we briefly survey the main characteristics of single-agent planning approaches and MAP for non-strategic agents (left branch in Figure 2.1).

2.1.1 Classical Single-Agent Planning

In a classical planning model (Ghallab, Nau, et al. 2004), the world is represented through a finite set of *states* which are *fully observable* by the planning entity. The world is deterministic, which means that the application of an action over a state leads deterministically to a single other state. The world is also static and hence, the state of the world does not change until an action is applied. The planning process is carried out offline, so a planner does not consider external changes that occur in the world. The actions have no duration and numeric reasoning is not considered. Finally, the goals to achieve are explicit and immutable. Even considering all these assumptions, this classical form of domain-independent single-agent planning is PSPACE-complete (Bylander 1994).

In classical planning, a state of the world is defined through a finite set of facts or *literals* that represents a situation of the world. A literal is an atom

composed of a predicate symbol and a possible empty finite set of parameters that denote objects of the world. The states of the world change through the application of the *planning actions*. In general, a planning action is defined as a set of *preconditions* that must be satisfied in a world state for the action to be applicable in such state, and a set of *effects* which transform the world state into a new one by adding and/or deleting literals. Hence, a planning task is defined as $\mathcal{T} = \langle \mathcal{I}, \mathcal{A}, \mathcal{G} \rangle$, where $\mathcal{I}$ is the *initial state* of the world, $\mathcal{A}$ is the set of actions, and $\mathcal{G}$ is the set of goals. A solution to a planning task $\mathcal{T}$ is a plan or course of actions from the set $\mathcal{A}$ that when applied in $\mathcal{I}$ leads to a state in which the goals $\mathcal{G}$ are achieved.

An important aspect in planning is how to represent all the components of a task with an expressive language. One of the first planning languages is STRIPS[1] (Fikes et al. 1971), which has influenced most of the existing planners. STRIPS is a compact and simple model to specify planning domains. The most widely extended version of STRIPS is the Planning Domain Definition Language (PDDL) (Ghallab, Howe, et al. 1998), which has become the standard language within the planning community. There exist several versions of PDDL: PDDL2.1 introduces time management and numeric capabilities (Fox et al. 2003); PDDL2.2 adds derived actions and timed initial literals (Edelkamp 2003); and PDDL3.0 (Gerevini and Long 2005) introduces preferences, soft constraints and state trajectory constraints. The latest version is PDDL3.1 (Kovacs 2011) that enriches the language with *SAS+*-like task representations (Bäckström and Nebel 1995). PDDL3.1 introduces object fluents; i.e., state variables that are neither binary (true/false) nor numeric (real-valued), but instead are mapped to a finite domain of objects. This representation is mainly inspired by the Functional Strips formalism (Geffner 2000).

[1] Stanford Research Institute Problem Solver (STRIPS) language.

There is a large variety of classical single-agent planners, many of which have participated in the different International Planning Competitions (IPCs)[2] held to date. Single-agent planning systems are usually classified according to the search space they explore and the direction of the search (Ghallab, Nau, et al. 2004).

Heuristic planning is one the most popular planning paradigms nowadays. The key of the success of heuristic planners is the design of powerful state-based heuristics. The Heuristic Search Planner (HSP) (Bonet et al. 2001) is one of the first state-based systems which uses domain-independent heuristic search. The additive heuristic of HSP is defined as the sum of costs of the individual goals in $\mathcal{G}$, where the cost of a single atom is estimated by considering a relaxed planning task in which all delete lists of the actions are ignored. The Fast Forward (FF) planning system (Hoffmann et al. 2001) uses the relaxed plan heuristic h_{FF}, which is defined as the number of actions of a plan that solves the relaxed planning task. FF works with a Enforced Hill Climbing search, an strategy that searches exhaustively for nodes with a better heuristic value than the previous best node. On the other hand, the popular Fast Downward (FD) planner (Helmert 2006) introduces *SAS+* planning representations. For each state variable, FD infers its associated Domain Transition Graph (DTG), a structure that captures the evolution of the value of a variable through the application of the actions. The heuristic function h_{DTG} of FD uses the DTG of the variables to estimate the distance or number of actions that are required for a variable to reach a final value v_f from an initial value v_i. FD alternates h_{DTG} and h_{FF}.

Most recent solvers draw upon the h_{FF} heuristic and the FD planner due to its remarkable performance. For instance, some significant optimal planners are Fast Downward Stone Soup-1 (Helmert, Röger, et al. 2011), Selective Max

[2] http://icaps-conference.org/index.php/Main/Competitions

(Domshlak et al. 2010) and Merge&Shrink (Helmert, Haslum, et al. 2014). The LAMA satisficing planner (Richter et al. 2010) is a FD-based planner that applies *landmarks* (facts that must hold at some point in every solution of a planning task) to enhance heuristic search. LAMA reuses the multi-heuristic search strategy of FD to alternate a landmark-based estimator and a variant of the h_{FF} heuristic.

In the last IPC-8 (Vallati et al. 2015) celebrated in 2014, 29 planners out of 67 in the deterministic track were built on top of the FD planning system (Helmert 2006), including the winner of the sequential optimal track, SymBA*-2 (Torralba et al. 2016). Similarly, the winner of the sequential multi-core track, ArvandHerd (Valenzano et al. 2012), is a portfolio-based approach that combines random-walk and best-first search by simultaneously using LAMA (Richter et al. 2010) and Arvand (Nakhost et al. 2009) planners. IBaCoP2 (Cenamor et al. 2014) is another portfolio approach that was the winner of the sequential satisficing track. This portfolio approach combines all 27 planners from the sequential satisficing track of the IPC-7 plus LPG-td (Gerevini and Serina 2002). Particularly, IBaCoP2 uses a classification model to configure the planners combinations with predictive models. Therefore, it seems portfolio approaches, which combine multiple planners to take advantage of their strengths in different problems, is becoming an important trend in planning since a significant number of these approaches were presented in the 2014 IPC-8 (Vallati et al. 2015).

Despite the success and great advances in heuristic planning, these planners are limited to sequential plans, which has also motivated a great deal of investigation in **Partial-Order Planning (POP)** (Penberthy et al. 1992; Barrett et al. 1994). The POP paradigm introduces an approach where the planner maintains partial-order relationships between actions without establishing a particular order between every pair of actions. POP is based on the *least*

commitment principle (Weld 1994) which determines that decisions about the order of the actions and the value of the variables are deferred as long as possible during the search process. The original POP formalism is designed as a regressive search process.

Some recent works have proposed a reformulation of the classical regressive POP algorithm into a forward-chaining search strategy (Coles et al. 2010; Younes et al. 2003). This allows combining a partial-order reasoning with an explicit representation of states. The main advantage of applying a forward POP search is a critical turning point that permits these methods to effectively apply state-based heuristic functions, which has given rise to remarkably efficient POP systems. Nowadays, planners that generate partial-order plans are basically found in temporal planning like SGPlan (Chen, Wah, et al. 2006), Temporal Fast Downward (Eyerich et al. 2012), $D_A E_{YAHSP}$ (Khouadjia et al. 2013), YAHSP2 (Vidal 2011), POPF2 (Coles et al. 2010), OPTIC (Benton, Coles, et al. 2012) and FLAP (Sapena et al. 2014).

It is also worth mentioning the **Hierarchical Task Network (HTN)** planning paradigm, a formalism that solves a planning task by applying a successive goal decomposition (Georgievski et al. 2015). The objective of an HTN planner (Erol et al. 1994) is to perform a set of HTN tasks, being this a primitive action or a task which is decomposable into smaller tasks. The input of an HTN planner includes a set of actions and a set of methods to indicate how a task can be decomposed. Hence, HTN progressively decomposes tasks until only primitive or executable actions remain. HTN is widely used in practical applications like decision support in forest fire (De La Asunción et al. 2005), or generation of clinical treatment plans (Sánchez-Garzón et al. 2013; Fdez-Olivares et al. 2011). Some of the most significant HTN planners are SHOP (Nau, Cao, et al. 1999), its successor SHOP2 (Nau, Au, et al. 2003), and SIADEX (Castillo et al. 2005).

2.1.2 Multi-Agent Planning

A *cooperative MAP* task is defined as the collective effort of multiple agents towards achieving a common goal. In the last few years, there has been significant research in the field of cooperative MAP, partly motivated by the 2015 Competition of Distributed and Multi-Agent Planners (CoDMAP)[3] (Komenda et al. 2016). Cooperative MAP is applied to solve tasks which cannot be solved by a single agent or that are better solved by cooperating (Durfee 2001). In the former case, combining the abilities of the agents is needed in order to solve a common task. For example, a task that involves various agents which are spatially distributed (located in different geographical areas) or functionally distributed (agents do not all have the same abilities) and some or all of the agents are required to solve the planning task (Torreño et al. 2014b).

Privacy is one of the main motivations to adopt a MAP approach. Privacy means coordinating agents without making sensitive information publicly available (Tožička, Jakubův, Komenda, and Pěchouček 2016). Whereas this aspect was initially neglected in former MAP solvers, the most recent approaches tackle this issue through the development of robust privacy-preserving algorithms (Štolba, Tožička, et al. 2016). The most common method is the obfuscation to occlude private information.

Among the existing MAP languages, Multi-Agent Planning Domain Definition Language (MA-PDDL)[4] stands out as the first attempt to create a standard specification language for cooperative MAP tasks. MA-PDDL extends and adapts PDDL3.1 (Kovacs 2011) to a MAP context, and allows for factored and unfactored task representations. In a factored input, each agent receives the

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

[4] Please refer to http://agents.fel.cvut.cz/codmap/MA-PDDL-BNF-20150221.pdf for a complete definition of the syntax of MA-PDDL.

specification of its planning task, which includes the knowledge and information the agent has on the task as well as the set of task goals which are shared by all of the agents. Additionally, the representation of the individual tasks includes information regarding agents' privacy so as to define the data which cannot be shared with other agents.

A large number of the planners that participated in the centralized track of the 2015 CoDMAP compile a MAP task into a set of single-agent tasks. The Agent Decomposition-based Planner (ADP) (Crosby, Rovatsos, and Petrick 2013), which was the winner of this track, uses a fully automated process that inspects the multi-agent nature of the planning task and calculates an agent decomposition that results in a set of decoupled local tasks. Then, ADP applies a state-based centralized planning procedure to solve the MAP task. Similarly, CMAP (Borrajo and Fernández 2015), MAPR (Borrajo 2013) and PMR (Luis et al. 2014) follow a goal-allocation mechanism and obfuscation strategy to solve the MAP task with single-agent planning technology. MAP-LAPKT (Muise et al. 2015) conceives a MAP task as a problem that can be transformed and solved by a single-agent planner using the appropriate encoding. MADLA (Štolba and Komenda 2015) is a centralized solver that runs one thread per agent on a single machine and combines two versions of the h_{FF} heuristic.

Other planners that apply a distributed agent search to solve the MAP task participated in the distributed track of the CoDMAP. PSM (Tožička, Jakubův, and Komenda 2015) follows a compact representation of local agents plans into Finite Automata, called Planning State Machines (PSMs). This planner makes public projections of PSMs and merges the public parts of individual PSMs until obtaining a solution to the MAP task. PSM was the top performer of this track, which was partly motivated by its efficient handling of agent communications. MAPlan (Fišer et al. 2015) is a distributed approach that implements a collection of state-space search methods and several local and

global heuristic functions. This planner can be used in a single-machine or in a distributed fashion by means of network message passing. Multi-Heuristic Forward Multi-Agent Planning (MH-FMAP) (Torreño, Sapena, et al. 2015) is a forward-chaining POP solver in which agents apply a distributed exploration of the plan space. Agents locally compute plans through an embedded POP component and they build a joint search tree by following an A* search scheme guided by global heuristic functions.

The cooperative MAP research community has gained importance in the last years and has established the base to attain MAP tasks through cooperation. MAP, however, has also been an active research field within the Multi-Agent System (MAS) community. The investigation of multi-agent planning in MAS is principally motivated by the distributed nature of the tasks and systems, and put the focus on the design of strategies to coordinate the plans of multiple decentralized agents. This activity, known as *decentralized planning*, is mostly concerned with the coordination of local plans of the agents. Therefore, in contrast to cooperative MAP, agents in this setting perform planning independently and they do not directly cooperate to achieve their goals.

One of the most representative approaches to decentralized planning is PGP (Durfee and Lesser 1991), a domain-specific approach where agents build their partial global view of the planning problem. The search algorithm of PGP finds local plans in the agents' plan-space that are coordinated to meet the goals of all the agents. Generalized PGP (GPGP) is a domain-independent extension of PGP (Decker et al. 1992; Lesser et al. 2004) that separates the process of coordination from local scheduling, which allows agents to reduce communications by sharing more abstract information.

In a plan-merging process, agents achieve their goals more efficiently by joining their efforts although this generates some bindings among the agents'

plans. For instance, the planner proposed in (desJardins and Wolverton 1999) defines a communication scheme where agents create partial views of sub-plans, and then, an agent performs a centralized plan merging process to solve the threats or conflicts. Similarly, other local-plan merging strategies have been proposed in (Ephrati et al. 1995; De Weerdt et al. 2003; Cox et al. 2004).

Decentralized planning is also focused on avoiding interactions between the agents' plans by means of coordination methods (Boutilier et al. 2001), or using distributed constraint optimization techniques (Cox et al. 2009).

In general, there has been a rather intensive research in decentralized planning related to the task of combining the local plans of the agents into a global solution. Nevertheless, in planning scenarios where agents feature private interests, the techniques of cooperative MAP or decentralized planning are not suitable as the strategic behavior of agents is ignored.

2.2 Game-theoretic Planning Approaches

A *rational self-interested agent* is an entity which has its own description of the states of the world it prefers, so that its behavior is motivated by this description (Shoham et al. 2009). The mathematical study of interaction between rational self-interested agents is called *game theory* (Von Neumann and Morgenstern 2007; Myerson 2013; Osborne et al. 1994). The key issue in game theory is that individuals pursue their own interests.

Coalitional or Cooperative Game Theory (CGT) has teams as the central unit, rather than agents, and it is devoted to analyzing how agents work together by forming coalitions (Brandenburger 2007). Specifically, CGT predicts which coalition agents will form and hence the utility that agents will obtain. On the

other hand, Non-Cooperative Game Theory (NCGT) is concerned with strategic equilibrium and individual utility maximization given the actions of other people. Practically speaking, NCGT deals with various equilibrium concepts and it is based on a precise description of the game in question.

Within NCGT, we can distinguish between strictly competitive and non-strictly competitive games. On the one hand, strictly competitive agents have opposed interests, and hence, the goal of an agent is to defeat or harm the other agents. This setting is represented by the so-called zero-sum games (Von Neumann 1928; Von Neumann and Morgenstern 2007). Two-player board games are typically deployed in a strictly competitive setting such as Chess, Checkers, Go, and so on, where a joint strategy that is better for one player is worse for the other player. This formalizes the intuition that the interests of players are diametrically opposed. On the other hand, non-cooperative games that take place in non-strictly competitive settings feature agents that have some complementary interests and some interests that may be completely opposed. This type of games are defined as non-zero-sum games, also called general-sum games, where the winnings and losses of all agents do not add up to zero and win-win situations can be reached (Gillies 1959; Shoham et al. 2009, Chapter 3).

In a planning setting, cooperative game theory is named *coalitional MAP*. Similarly, non-cooperative game theory with strictly competitive agents is called *adversarial MAP*. Finally, as we pointed out in the Introduction of this PhD thesis, the approaches that feature non-strictly competitive planning agents will be referred to as *non-cooperative MAP*.

In this section, we briefly analyze the planning approaches that feature self-interested agents (right branch of Figure 2.1). We firstly present the most

relevant approaches of coalitional MAP and adversarial MAP. Afterwards, we present an overview of non-cooperative MAP approaches.

2.2.1 Coalitional MAP

CGT considers groups of agents which benefit by forming coalitions (Brandenburger 2007). For instance, coalitional resource games are games in which agents cooperate by joining resources to achieve mutually satisfying goals (Dunne et al. 2010). This work explores equilibrium concepts like the core (Banerjee et al. 2001) and Subgame Perfect Equilibrium (SPE) (Selten 1975), and a negotiation strategy to form coalitions. Additionally, the work in (Shehory et al. 1998) proposes methods for task allocation through coalition formation among agents. These methods are applied in situations where agents study which coalitions to form in order to achieve more goals.

CGT has been applied in planning scenarios to solve coalitional goal allocation problems. The Coalition-Planning Game (CoPG) proposal (Brafman et al. 2009) extends STRIPS-like models of single-agent planning to a system of multiple self-interested agents that may need to cooperate in order to achieve their single associated subgoals, but are assumed to do so only in order to increase their net benefit. The second approximation of this work is the Auction-Planning Game (AuPG), where coalitions of agents compete for the achievement of a single goal which yields a monetary reward for the coalition. The profit is always distributed among the coalition agents.

A later work in the line of CoPG was presented in (Crosby and Rovatsos 2011). In this case, the proposed approach solves the so-called “safe” CoPG, a subset of the CoPGs where no agent can benefit from making another agent’s plan invalid. A single-agent planner based on the h_{FF} heuristic is used to solve the multi-agent problem.

More recently, Hadad et al. 2013 presented a temporal reasoning mechanism for self-interested planning agents. In this work, agents' behavior is modeled on the basis of the Belief-Desire-Intention (BDI) theoretical model of cooperation to compute joint plans with time constraints. The mechanism ensures temporal consistency of a cooperative plan and it has been tested in real-life scenarios.

2.2.2 Adversarial MAP

NCGT in a strictly competitive setting studies problems in which the self-interested agents have opposed goals. These games are known as zero-sum games (Von Neumann 1928; Von Neumann and Morgenstern 2007). Some popular examples are two-player board games, the *matching pennies* game (Mookherjee et al. 1994; Ochs 1995), or Rochambeau[5] (also known as Rock, Paper, Scissors), which provides a three-strategy generalization of the matching pennies game. In a planning setting, a strictly competitive game, named adversarial MAP, is interpreted as agents pursuing completely opposed goals, e.g., one agent has to achieve the goal g and another one has to achieve $\neg g$.

Some adversarial MAP approaches focus on solving problems in non-deterministic and unpredictable scenarios (Applegate et al. 1990). In this work, authors present an architecture for adversarial planning in battle management that involves control of several semi-autonomous intelligent agents; the need to adjust plans dynamically according to developments during plan execution; and the need to consider the presence of an adversary in devising plans. (Jensen et al. 2001) presents universal adversarial MAP algorithms for non-deterministic finite domains. These algorithms extend the family of ordered binary decision diagrams and are applied to stochastic games.

[5] http://www.rpscontest.com/

Another research line of adversarial MAP is devoted to board games or general video games. (Sailer et al. 2007) propose a planning framework that uses strategy simulation to achieve Nash equilibrium solutions. This framework is applied to army deployment problems in real-time strategy settings in order to improve strategic behavior of agents in modern video games. The works in (Willmott et al. 1998; Willmott et al. 2001) present an adversarial HTN planning framework for goal-directed game playing. Particularly, this approach is only applied to solve the well-known game of Go[6].

Similarly to alpha-beta pruning (Knuth et al. 1975), (Bercher et al. 2008) propose a forward-chaining approach to adversarial MAP based on the AND/OR* algorithm, which is guided by a heuristic evaluation function inspired by the relaxed planning graph used in the calculation of the heuristic function h_{FF}. This approach is only applicable to two-player games.

In summary, in strictly competitive scenarios agents have opposed goals, they seek to harm each other and only care about decisions that benefit them.

2.2.3 Non-cooperative MAP

Agents in a non-strictly competitive setting do not seek to harm each other but, since they have complementary and contrary interests, their purpose is to apply a collaborative strategy and conflict resolution process that aims to accommodate all participants (win-win strategy). In this section, we first introduce the type of problem we aim to solve with a typical example of general-sum games (Myerson 2013; Osborne et al. 1994). Additionally, we show the mapping between the players' strategies of a game and the agents' plans in a planning context. Subsequently, we present the related work in non-cooperative MAP to solve this type of problems and, finally, we introduce the main features

[6] http://www.usgo.org/way-go

of the models that will be presented in this PhD thesis in order to compare them with the approaches in the related work.

Let us consider the well-known coordination game "the Battle of the Sexes" shown in the payoff matrix of Table 2.1 (Shoham et al. 2009, Chapter 3) (Luce et al. 1957, Chapter 5). The first number of each cell represents the utility obtained by the row player given the combination of strategies of that cell. Similarly, the second number is the utility that the column player obtains given the combination of strategies represented in the cell. In this game, two players must decide where to go, to a rock music concert or to a pop music concert. Each player would prefer to go to a different concert; that is, row player prefers the rock music concert, and column player prefers the pop music concert. However, both would prefer going to the same concert together rather than to different ones, what explains that the utilities shown in Table 2.1 are 0 when each player goes to a different concert and over 0 when they attend together to the same event. This is a general-sum game where the self-interested agents coordinate their strategies so as to obtain the maximum possible utility.

Table 2.1: Two-player coordination game in normal-form

	Rock	Pop
Rock	**3, 2**	0, 0
Pop	0, 0	**2, 3**

Let us now consider the strategies of the players as plans. From a planning perspective, we assume the existence of a *conflict* when the players attend different concerts, and hence the utilities for both are 0. In case both players would go to the same concert, we assume that the player who goes to its non-preferred concert will receive lower utility because the plan is more costly to

perform. Two different Nash Equilibria (NE) (Nash 1951) can be obtained in this game, namely, the combination of the strategies *Rock*, *Rock*, with utilities (3, 2), or the combination of the strategies *Pop*, *Pop*, with utilities (2, 3). These are the only stable outcomes from which agents would never deviate because otherwise it would imply a utility loss.

In non-cooperative MAP, agents do not seek help from others to satisfy their goals; i.e., they do not form coalitions with the purpose of building their plans. There are several types of problems in non-cooperative MAP. In some problems, agents have to solve a MAP task and goal allocation is applied to distribute the goals of the task while retaining the agents' private interests. Other problems feature multiple individual planning tasks. In this case, agents are assumed to independently work on their own part of the planning problem, and then, their plans have to be coordinated with others'. Similar settings focus on determining the amount of goals each agent can solve of its own planning task (soft goals), depending on the interactions with others', through non-cooperative games.

Non-cooperative MAP settings have been used for goal allocation (Nissim and Brafman 2013). Agents have to obtain an optimal solution to a MAP task, which means they have a common interest, while satisfying their private incentives. The objective is to determine the goals of the MAP task that will be solved by each agent, while guaranteeing an optimal solution that maximizes the sum of the agents' utilities, i.e., utilitarian social welfare. Hence, when an agent is bidding for a goal, a payment is applied to reflect the impact of each agent's participation on the other agents. This problem is solved with the Vickrey–Clarke–Groves (VCG) mechanism (Vickrey 1961; Clarke 1971; Groves 1973). Other approaches use similar auction mechanisms to distribute goals among agents (Van Der Krogt et al. 2005). In this work, agents bid for the

goals to solve and the MAP task is tackled with single-agent plan repair systems.

MAP with self-interested agents is commonly regarded as a coordination problem in which several agents interact. Some approaches apply pre-planning coordination and decompose a global task into individual sub-tasks that agents can solve independently (Buzing et al. 2006). The underlying goal is to find a minimal set of precedence constraints that guarantees autonomous planning. The complexity of pre-planning coordination yields the problems intractable, however, reasonable solutions can be obtained by adding some additional constraints (Mors et al. 2005), or using approximation algorithms (Buzing et al. 2006). A recent work in (Hrnčíř et al. 2015) presents an application for ridesharing aimed at finding routes that travelers can share in order to save costs. The solution proposed by this work ensures that each individual is better off taking the shared ride rather than traveling alone.

In some MAP environments with multiple individual planning tasks, the aim is to determine the goals that each agent solves (soft goals) depending on the negative interactions with other agents. In (Galuszka et al. 2010), the authors propose a STRIPS model that calculates a solution plan by doing the inverse of the problem (the initial states of the agents are translated to soft goals, and the original goal of each agent to its initial state). This model represents the conflicts between the actions of the agents' plans through a payoff matrix and finds an equilibrium for the whole problem goals or a part of them.

In (Bowling et al. 2003) a preliminary formalization of equilibrium in multi-agent planning is introduced. MAP solutions are classified according to the agents' possibility of reaching their goals and the paths of execution (combinations of local plans). However, this work has not been completely developed or empirically tested. Similarly, the work in (Larbi et al. 2007) extends the classi-

cal planning model to multi-agent scenarios where agents perform online planning. Each agent wants to achieve its goals, but since planning is interleaved with execution, the feasibility of its actions is uncertain. Hence, agents decide which action to apply at each time step through a non-cooperative game. A "satisfaction profile" is defined to determine the possibilities that an agent has to achieve its goals when applying each action.

Best-Response Planning (BRP) (Jonsson et al. 2011) is an approach specifically devoted to solving the so-called *congestion games* (Rosenthal 1973), where the simultaneous use of a resource by multiple agents increases its cost. BRP is also applied to some planning domains for self-interested agents where convergence to equilibrium solutions is not guaranteed. In BRP, an initial conflict-free joint plan is calculated with the *DisCSP* cooperative planner (Nissim, Brafman, and Domshlak 2010). Then, best-response dynamics, an iterative process in which each agent proposes its best plan considering the other agents plans, are applied in order to improve the initial solution. BRP is able to synthesize stable joint plans, where all agents achieve their goals, with remarkable performance in the presented experimental results with planning domains adapted to self-interested agents.

Non-cooperative MAP tasks can be seen as the coordination of the agents' plans to come up with an executable joint plan. Agents have both cooperative interests (all agents want their plans fit together so as to ensure they are executable) as well as contradictory interests (every agent wants the execution of its plan to prevail over the others' in case of conflict). Since agents are self-interested, stable (equilibrium) solution joint plans must be guaranteed so that no agent will deviate from the solution. We can distinguish two different views to tackle non-cooperative MAP tasks aimed at synthesizing stable joint plans.

Agents in (Jordán et al. 2015) (FENOCOP, Chapter 3) have a limited set of precomputed plans that solve their planning tasks and the MAP task of synthesizing a stable joint plan is solved with a two-game approach. One game determines the plan that each agent will use by computing an equilibrium. Each joint plan is scheduled through a second game where agents avoid planning conflicts by delaying the execution of their actions. Particularly, this game computes equilibrium solutions that are also Pareto Optimal (PO) (Arrow 1963) and fair, thus satisfying agents as much as possible. This is a significant novelty in the non-cooperative MAP literature. However, this approach does not consider congestion issues, planning conflicts entail $-\infty$ utility for all agents, and the calculation of fair PO equilibrium solutions is a difficult task to solve in larger problems.

It is crucial to use more efficient methods to deal with self-interested agents in order to solve larger problems than in (Jordán et al. 2015). Additionally, considering congestion issues and planning conflicts as part of the agents' cost yields a more realistic approach than BRP (Jonsson et al. 2011). Better-response dynamics can be used in a similar setting as BRP, thus providing planning agents with an unlimited amount of plans, i.e., planning capabilities in a MAP context. In this way, better-response dynamics with planning agents allows them to perform an iterative process to find a stable joint plan, while strategically avoiding conflicts and congestions. Moreover, the need of an initial conflict-free joint plan provided by a cooperative planner in BRP must be avoided since it would generate synergies between the agents, which makes no sense in a non-cooperative setting. Finally, it is necessary to study under which conditions convergence to an equilibrium is guaranteed in non-cooperative MAP tasks, as well as the theoretical complexity. All these aspects are attained in our BRPS model (Chapter 5).

Table 2.2: Feature comparison of different approaches

	BRP	FENOCOP	BRPS
Unlimited Plans	**Yes**	No	**Yes**
Compute All NE	No	**Yes**	No
Pareto optimal	No	**Yes**	No
Fairness	No	**Yes**	No
Congestion	**Yes**	No	**Yes**
Conflicts as cost	No	No	**Yes**
Joint Plan from scratch	No	**Yes**	**Yes**
Complexity	NP-hard	NP-hard	**PLS-hard**

Table 2.2 presents a comparison of the main features of BRP, and the two models that we propose in this PhD thesis. All these approaches can solve non-cooperative MAP problems, however, there are significant differences between them. We wish to highlight that FENOCOP is the only approach that obtains PO and fair solutions among the computed NE. However, it does not have an unlimited set of plans (i.e., planning capabilities to generate new plans, but not infinite), which may restrict the amount of possible solution plans, and it does not consider congestion issues. The BRPS approach is able to synthesize joint plans from scratch while considering congestion and planning conflicts in the agents' cost functions. This is an important feature that provides the agents with a more realistic behavior to tackle with real-world non-cooperative MAP problems. Additionally, BRPS has promising complexity results under some assumptions.

Every non-cooperative MAP problem can be represented by an equivalent non-cooperative game. The strategies of the game are the plans of the non-cooperative MAP problem. However, computing plans is a hard task, and

hence, this complicates the problem of computing equilibrium solutions since the strategies in our games are plans or schedules that have to be generated. Finally, the planning perspective is adding to non-cooperative games a realistic approach to solve real world problems where strategic planning is required.

2.3 Conclusions

In general, there has been rather intensive research in planning and MAP without self-interested agents. Additionally, there has been some research in MAP approaches that feature self-interested agents such as coalitional MAP and adversarial MAP. However, the combination of game theory with MAP in a non-cooperative but not strictly competitive setting has been traditionally neglected by the research community. In this sense, there are only a few approaches which explore the impact of non-cooperative agents in a planning setting.

Therefore, we find necessary to fill the gap and explore what can be done in a new research line which we called non-cooperative MAP. We are convinced that as well as cooperative MAP has become an important field in the multi-agent systems and planning communities, a step forward must be done in order to cover an important part of real-life MAP problems in which the implied entities are represented by self-interested agents. Hence, game-theoretic solution concepts, as well as MAP techniques must be applied to synthesize stable joint plans that efficiently solve non-cooperative MAP tasks.

Chapter 3

FENOCOP: Fair Equilibria in Non-Cooperative Planning

Among the many variants of problems studied in the area of Multi-Agent Planning (MAP), in this chapter we focus on non-cooperative MAP tasks. One first consideration in this type of tasks is that agents are self-interested and seek their own benefit. Secondly, agents independently synthesize their own plan to reach their goals autonomously. Finally, since agents intend to execute their plans in a shared environment, and thus, interactions that negatively affect plans may arise, agents are willing to coordinate their plans in order to avoid the potential planning conflicts during a joint execution.

Consider, for instance, several truck drivers, each one in charge of delivering cargo. Interactions among their plans will appear if some of the truck drivers simultaneously attempt to access one depot with capacity limitations, or if a truck driver is unable to unload the cargo because the depot is already full.

When agents act in their own interest, an agent may not stick to the agreed plan if it has a better strategy, even at the cost of others. *Stable* outcomes are needed to guarantee that self-interested agents do not deviate from the agreed joint plan. We are also interested in outcomes that are *fair*, in order to balance out the individual satisfaction of the agents. Therefore, we look for stable outcomes that may also be good for all agents, i.e., strategies that are Nash Equilibrium (NE) (Nash 1951), Pareto Optimal (PO) (Arrow 1963), and fair. An outcome which is NE, PO and fair is desirable because no agent will be willing to deviate from an equilibrium while Pareto optimality removes any Pareto inefficient NE, and fairness guarantees a balance between the agents' utilities.

The existing approaches in the research line of non-cooperative MAP like (Larbi et al. 2007) and (Jonsson et al. 2011) do not properly synthesize a stable joint plan and they do not consider Pareto optimality and fairness. In (Kameda et al. 2012) two fairness measures are proposed in resource allocation called *Nash equilibrium based fair* and *Nash-proportionately fair*. However, these measures are proportionately based on a PO NE, which means that are attached to a particular proportionality of any of the PO NE outcomes. In addition, that work is also in the context of resource allocation and not related to MAP.

Non-cooperative MAP for self-interested agents is a problem in which planning and coordination are needed. Solving the whole planning problem is too complex because it is PSPACE-hard (Bylander 1994) and its runtime grows exponentially with the problem size. Since agents can solve their goals individually, it is reasonable to solve each individual planning problem separately an then coordinate them all by delaying conflicting actions in order to obtain a feasible joint plan. Conflicts may arise because of mutually exclusive actions

or because of the lack of applicability of an action in a planning stage. In this chapter, we thoroughly investigate the properties of this approach.

The problem we aim to solve is as follows: we consider a group of agents, each having one or more plans that attain its goals. Executing a particular plan reports the agent a utility that depends on the makespan (finish time) of the plan. Agents operate in a common environment, what may cause interactions between the agents' plans, thus preventing a concurrent execution. Each agent is willing to execute the plan that maximizes its utility but it does not know about the strategies of the remaining agents, how the plans will integrate with each other, or the impact of coordination on its utility.

This raises two problems which we tackle with a dual-game proposal, named Fair Equilibria in NOn-COoperative Planning (FENOCOP), thus splitting the complexity of the problem. On the one hand, agents must strategically decide which joint plan or plan profile (combination of one plan per agent) to execute. Each agent must select the plan that maximizes its utility, which depends on the initial utility of the plan and how the plan is scheduled. This is attained by the *General Game (GG)*.

On the other hand, given a plan profile, agents schedule their plans to avoid conflicts, obtaining a set of executable schedule profiles as a result. Among these schedule profiles, a fair, Pareto optimal and NE outcome is selected in order to keep agents as satisfied as possible. This problem is solved with the *Scheduling Game (SG)*.

The non-cooperative MAP problem we aim to solve with FENOCOP could be solved through a single-game approach. However, this would present several drawbacks. On the one hand, combining all the possible schedules of the agents' plans for all the plan profiles is a computationally hard task. We initially tested and discarded this single-game approach due to its high complex-

ity. On the other hand, it makes sense to divide FENOCOP into the GG and the SG, since they solve problems of different nature. The GG is used to choose which plan to execute, and the SG determines the best possible schedule for each combination of the agents' plans. Additionally, the SG can be used as an independent framework and the GG can be used with different SG algorithms, depending on the features of the problem. All in all, the separation of FENOCOP in two problems makes our approach more versatile.

As it is usually assumed in game theory, the structure of the game, the rationality of the agents and the payoffs that each agent receives are common knowledge. Agents do not have private information since their plans and schedules are known to the other agents in both the GG and the SG.

The solutions of the GG are always NE, and the solution of each SG meets three criteria: it is a NE, PO and fair joint plan. The idea of combining these two games was first presented in our work in (Jordán et al. 2015). All in all, our whole proposal, FENOCOP, contributes with several novelties:

- A general framework, called FENOCOP, that solves non-cooperative MAP tasks for independent agents that plan autonomously. Agents calculate a set of individual plans that solve their respective problems, and then engage in a game to select a plan schedule that allows them to execute their plans simultaneously in a common environment.
- A Scheduling Game that returns a NE, because an equilibrium concept is needed to avoid that any self-interested agent deviates unilaterally, PO, thus improving Pareto inefficient NE, and fair, which guarantees that the least satisfied agent is as satisfied as possible.
- Three different algorithms that solve the SG. Two algorithms which ensure that the solution of the SG is Pareto optimal and fair, while the third

algorithm, which obtains a Subgame Perfect Equilibrium (SPE) (Selten 1975), does not guarantee Pareto optimality nor fairness.

- A mechanism to explicitly handling conflicts among agents' actions and to update the plan utility according to the penalty. This is precisely the objective of the SG and the key contribution that makes our model a realistic approach to MAP with self-interested agents.

This chapter is structured as follows. Section 3.1 introduces the formal notions related to the planning task of the agents. Section 3.2 presents an overview of FENOCOP. Sections 3.3 and 3.4 outline the characteristics of the two-level game approach; the top-level General Game and the internal Scheduling Game. Section 3.5 is devoted to explain the algorithms for solving the Scheduling Game, and finally, we give some conclusions of this work in Section 3.6.

3.1 Planning Scenario

The problem we want to solve involves a set of n rational and self-interested agents $\mathcal{AG} = \{1, \ldots, n\}$, where each agent $i \in \mathcal{AG}$ has an individual task, which is defined as follows:

Definition 3.1.1. ***Individual task of an agent****. The task of an agent $i \in \mathcal{AG}$ is a tuple $\mathcal{T}^i = \langle \mathcal{I}^i, \Gamma^i \rangle$, where $\mathcal{I}^i$ describes the initial state of the task, and $\Gamma^i = \{\pi_1^i, \ldots, \pi_l^i\}$ is a finite set of plans that attain $\mathcal{T}^i$.*

Our model is based on propositional STRIPS planning tasks. In this context, a plan $\pi^i \in \Gamma^i$ is defined as a sequence of *actions* $\pi^i = [a_0^i, \ldots, a_{m-1}^i]$. An action $a \in \pi^i$ is a triple $a = \langle pre(a), add(a), del(a) \rangle$: $pre(a)$ is the set of *preconditions* of a; $add(a)$ and $del(a)$ are two lists that denote the *positive* and *negative*

effects of a, respectively. A state S is defined as the set of facts or *literals* that describe a state of the world. An action a is executable in a state S if $pre(a) \subseteq S$. Executing a in a state S yields a new state S', such that $S' = S \setminus del(a) \cup add(a)$.

The execution of a particular plan $\pi^i \in \Gamma^i$ reports agent i a reward or utility. In this planning scenario, every agent $i \in \mathcal{AG}$ wishes to execute the plan of Γ^i that reports the maximal utility.

Definition 3.1.2. Plan profile. *A plan profile is a collection of one plan per agent denoted with the tuple* $\Pi = (\pi^1, \pi^2, \ldots, \pi^n)$*, where* $\pi^i \in \Gamma^i$ *represents the individual plan choice of agent* i*.*

The *actual utility* that a plan π^i reports to agent i depends on the concurrent execution of π^i with the rest of plans of the plan profile Π. Therefore, in this problem, the objective of an agent $i \in \mathcal{AG}$ is to select a plan π^i of Γ^i such that, when scheduled along with the rest of agents' choices in Π, it reports maximum utility to i.

Definition 3.1.3. Schedule of a plan. *The schedule of a plan* $\pi^i \in \Gamma^i$ *is a temporal sequence of actions that results from interleaving the actions in* π^i *with an arbitrary number of empty actions* $\perp$*. A plan schedule indicates the action of* π^i *to be executed at each time point.*

We will denote by $\Upsilon^i = \{\psi_0^i, \psi_1^i, \ldots, \psi_x^i, \ldots\}$ *the infinite set of all possible schedules of plan* π^i*. Given a particular schedule* ψ_x^i*, the* finish time *of the execution of* ψ_x^i *will be the time instant of the last action in* ψ_x^i*. In general, given two plan schedules* $\psi_x^i, \psi_{x+1}^i \in \Upsilon^i$*, the finish time of* ψ_x^i *is assumed to be prior or equal to the finish time of* ψ_{x+1}^i*. In the following, we will simply use the notation* ψ^i *to refer to any schedule of* Υ^i*.*

The ideal schedule of a plan $\pi^i = [a^i_0, \ldots, a^i_{m-1}]$, ψ^i_0, consists in executing a^i_0 in the state at $t = 0$ or initial state $\mathcal{I}^i$, and executing the subsequent actions of π^i at consecutive time instants. Thus, presumably, agent i will finish the execution of π^i at $t = m-1$, the time of the last action scheduled in ψ^i_0 (a^i_{m-1}). However, since agents execute their plans simultaneously in a common environment, *conflicts* that prevent agents from executing the ideal schedules of their preferred plans may arise. In case that a conflict compromises the ideal schedule ψ^i_0 of a plan π^i, agent i may select an alternative schedule, ψ^i_x, which will comprise a number of empty actions $\perp$ that will help solve the conflict. The introduction of empty actions obviously entails a delay in the finish time of the plan execution, which in turn entails a loss of utility. The purpose of delaying actions is to avoid conflicts and ensure the executability or feasibility of a plan schedule.

Example 3.1.1. *Given a plan $\pi^i = [a^i_0, a^i_1, a^i_2, a^i_3]$ of agent i, possible schedules for π^i are: $\psi^i_0 = (a^i_0, a^i_1, a^i_2, a^i_3)$, $\psi^i_1 = (\perp, a^i_0, a^i_1, a^i_2, a^i_3)$, $\psi^i_{10} = (a^i_0, a^i_1, \perp, a^i_2, \perp, a^i_3)$, $\psi^i_{19} = (a^i_0, a^i_1, \perp, a^i_2, \perp, \perp, a^i_3)$, etc. Particularly, ψ^i_0 is the earliest plan execution of π^i (finishing at $t = 3$); ψ^i_1 completes the execution of π^i at $t = 4$, ψ^i_{10} at $t = 5$ and ψ^i_{19} at $t = 6$.*

The ultimate objective of the agents in $\mathcal{AG}$ is to come up with a combination of plan schedules (one per agent's plan) that is jointly executable. Since the plan choices of the agents may affect each other's utilities, the model proposed in this chapter is a non-cooperative game-theoretic approach that solves the problem of finding a conflict-free (feasible) *schedule profile* which guarantees that the agents' plans of a plan profile Π are executable.

Definition 3.1.4. Schedule profile. *Given a plan profile $\Pi = (\pi^1, \pi^2, \ldots, \pi^n)$, a schedule profile of Π, s_Π, is a combination of one schedule per plan in Π; that is, $s_\Pi = (\psi^1, \psi^2, \ldots, \psi^n)$, $\psi^i \in \Upsilon^i$.*

A schedule profile $s_\Pi = (\psi^1, \psi^2, \ldots, \psi^n)$ induces a sequence of *joint actions*. A joint action is a tuple $A_t = \langle a^1, a^2, \ldots, a^n \rangle$, where a^i is the action of ψ^i scheduled at time instant t. In other words, A_t collects the actions of the plan schedules in s_Π (one action per agent in $\mathcal{AG}$) that agents intend to execute at time t.

Example 3.1.2. *Given a schedule profile $s_\Pi = (\psi^1, \psi^2, \psi^3)$, $A_t = \langle a_2^1, \perp, a_3^3 \rangle$ is the joint action to be executed at time t, where agent 1 wants to execute its action a_2^1, agent 2 executes the empty action and agent 3 executes its action a_3^3.*

Joint actions are applied over *joint states*. The initial joint state of the problem, $\mathcal{I}$, is defined as the union of the inital states of the agents in $\mathcal{AG}$; that is, $\mathcal{I} = \mathcal{I}^1 \cup \ldots \cup \mathcal{I}^n$. A joint action A_t is *executable* in a joint state S if no *conflict* arises at the time of executing the actions of A_t. We identify two types of conflicts in A_t:

- *Precondition conflict*. One condition for A_t to be executable in a joint state S is that $\forall a \in \mathcal{A}_t, pre(a) \subseteq S$. It may happen that the execution of a joint action prior to A_t leads to a joint state S where some precondition of an action a of A_t does not hold. In this case, we say a precondition conflict occurs and, consequently, A_t is non-executable.

- *Mutually exclusive (mutex) conflict*. This happens when two actions a and a' of A_t cannot be simultaneously executed at time t due to a mutex relationship as identified in the *GraphPlan* approach (Blum et al. 1997). Particularly, two actions a and a' are said to be mutex if:

  - They have *inconsistent effects*; i.e., $add(a) \cap del(a') \neq \emptyset$.

  - They *interfere* with each other; i.e., $pre(a) \cap del(a') \neq \emptyset$.

Hence, if none of the above conflicts appears in A_t, then we say A_t is executable. The result of applying an executable joint action $A_t = \langle a^1, a^2, \ldots, a^n \rangle$ in a joint state S is a new joint state $S' = S \backslash (\bigcup_{i=1}^{n} del(a^i)) \cup (\bigcup_{i=1}^{n} add(a^i))$. When A_t is not executable, this may be fixed by delaying the action(s) in conflict through the introduction of empty actions in the corresponding schedule profile.

Definition 3.1.5. Feasible (conflict-free) schedule profile. *A schedule profile* $s_\Pi = (\psi^1, \psi^2, \ldots, \psi^n)$ *is* feasible *if and only if every joint action* A_t *of* s_Π *is executable.*

Example 3.1.3. *Let us assume that two agents 1 and 2 want to execute the plan profile* $\Pi = (\pi^1 = [a_0^1, a_1^1, a_2^1], \pi^2 = [a_0^2, a_1^2, a_2^2])$*; a possible schedule profile is* $s_\Pi = (\psi^1 = (a_0^1, \bot, \bot, a_1^1, a_2^1), \psi^2 = (\bot, a_0^2, a_1^2, a_2^2))$*. Additionally,* s_Π *is a feasible schedule profile if every joint action is executable (the joint actions for* s_Π *are* $A_0 = \langle a_0^1, \bot \rangle$*,* $A_1 = \langle \bot, a_0^2 \rangle$*,* $A_2 = \langle \bot, a_1^2 \rangle$*,* $A_3 = \langle a_1^1, a_2^2 \rangle$*,* $A_4 = \langle a_2^1 \rangle$*).*

Given a plan profile $\Pi = (\pi^1, \ldots, \pi^n)$ and an associated schedule profile $s_\Pi = (\psi^1, \ldots, \psi^i, \ldots, \psi^n)$, the *maximum number of empty actions* in the schedule ψ^i of an agent i, is limited by the sum of the actions of the other agents' plans in Π, denoted by λ_Π^i:

$$\lambda_\Pi^i = \sum_{j \neq i}^{\pi^j \in \Pi} |\pi^j| \tag{3.1}$$

If we consider a problem where the number of schedules of a plan π^i associated to a plan profile Π is not limited by λ_Π^i, it is possible to find additional schedule profiles by adding more empty actions. Any additional schedule profile of a non-limited problem will report less utility to (at least) some agent i because it would include a number of empty actions larger than λ_Π^i. Therefore, we can conclude that the additional schedule profiles that can be formed in

a non-limited planning problem are weakly Pareto dominated by the schedule profiles of the original problem limited by λ_{Π}^{i}.

Example 3.1.4. *Given a plan profile* $\Pi = (\pi^1 = [a_0^1, a_1^1, a_2^1], \pi^2 = [a_0^2, a_1^2, a_2^2])$, $\lambda_{\Pi}^{i} = 3$ *for both agents,* $i = \{1, 2\}$. *A schedule with more than 3 empty actions for any agent is useless since the maximum number of empty actions necessary to address the conflicts is 3. For instance, the schedule profile* $s_{\Pi} = (\psi^1 = (a_0^1, a_1^1, a_2^1), \psi^2 = (\bot, \bot, \bot, a_0^2, a_1^2, a_2^2))$ *introduces 3 empty actions in* ψ^2 *and so all the joint actions in* s_{Π} *include a single action (*$A_0 = \langle a_0^1, \bot \rangle$, $A_1 = \langle a_1^1, \bot \rangle$, $A_2 = \langle a_2^1, \bot \rangle$, $A_3 = \langle \bot, a_0^2 \rangle$, $A_4 = \langle \bot, a_1^2 \rangle$ *and* $A_5 = \langle \bot, a_2^2 \rangle$*).*

Thus, given a plan profile Π*, if a feasible schedule profile cannot be obtained by means of* λ_{Π}^{i} *empty actions for every agent* $i \in AG$*, introducing more empty actions than* λ_{Π}^{i} *in the plan schedule of any agent will not yield a feasible schedule profile for* Π*. In this case, we say that all the schedule profiles for* Π *are* unfeasible*. Particularly, an unfeasible schedule profile is due to a precondition conflict because mutex conflicts are always solvable by introducing empty actions. However, even introducing* λ_{Π}^{i} *empty actions in all the schedule profiles, it may not be possible to find a joint state* S *in which all the preconditions of an action in a joint action* A_t *are satisfied.*

Definition 3.1.6. Utility of a plan schedule. *The utility function* $u^i : \Upsilon^i \rightarrow \mathbb{R}$ *returns the utility of a schedule* ψ^i *of a plan* π^i *for agent* i*. For a given* π^i*, the difference of utility of two plan schedules* ψ_x^i *and* $\psi_{x'}^i$*,* $x' > x$*, is given only by the difference in their finish execution times. The latter the finish time, the less utility. Consequently, by default, the ideal schedule* ψ_0^i *of a plan* π^i *is the schedule that reports agent* i *the maximal utility and the rest of schedules of* Υ^i *will have a lower utility accordingly to their finish time. An unfeasible (non-executable) schedule profile reports each agent* $i \in \mathcal{AG}$ *a utility* $u^i = -\infty$*.*

In this section, we have introduced and formalized all the components that are necessary for the specification of our game-theoretic approach FENOCOP.

3.2 Overview of FENOCOP

FENOCOP (Fair Equilibria in NOn-COoperative Planning) is our computational framework for the resolution of conflicts in non-cooperative MAP. As described in Section 3.1, the problem we aim to solve involves a set of self-interested planning agents, $\mathcal{AG}$, where each agent i independently works on its individual task $\mathcal{T}^i$ by calculating a finite collection Γ^i of plans (strategies) of different utility that solve $\mathcal{T}^i$.

Every agent i wishes to execute the ideal plan schedule ψ_0^i of the maximum utility plan π^i. On the other hand, given that this is a non-strictly competitive environment, agent i also wants to make its course of action ψ_0^i compatible with the rest of the agents' proposals of a plan profile and thus ensure that every agent is able to execute a plan that achieves its task.

Conflicts may appear when the plan schedules of multiple agents are put together to execution in a shared environment. A conflict between two particular plan schedules ψ_x^i and ψ_y^j entails that either agent i or agent j cannot execute its plan. When this happens, one or both agents must switch to a different schedule so as to avoid the interference. Assuming agent i selects a new schedule $\psi_{x'}^i$, some actions of ψ_x^i will be delayed in $\psi_{x'}^i$ through the inclusion of empty actions in order to solve the conflict, which in turn implies a delay in the finish time of the execution of agent i. If the new schedule $\psi_{x'}^i$ entails a significant loss of utility, agent i may select a different plan from Γ^i that, when scheduled with the rest of agents' plans, brings higher utility. Hence, agents

must find together a feasible schedule profile s_Π that ensures the executability of the plans while satisfying the private interests (utility) of the participants.

A rational way of solving the conflicts that arise among a set of self-interested agents with potentially conflicting interests implies modeling the problem as a non-cooperative game. FENOCOP is a non-cooperative dual-game mechanism guided by a top-level game called *General Game* (GG), which leverages an internal game called *Scheduling Game* (SG). Particularly, the GG of FENOCOP works as follows:

1. It generates the $\Gamma^1 \times \ldots \times \Gamma^n$ plan profiles that result from combining the strategies of the n agents in $\mathcal{AG}$.

2. For every plan profile Π, the GG calls to the SG to calculate a schedule profile s_Π. The outcome s_Π returned by the SG is a Nash Equilibrium (NE), Pareto-Optimal (PO) and fair solution. A solution that meets these properties is desirable because 1) it is a *stable outcome* from which no agent will be willing to deviate; 2) a PO outcome outperforms any Pareto-inefficient NE; and 3) a fair solution guarantees a balance among the agents utilities.

3. From the set of feasible or unfeasible schedule profiles $\{s_{\Pi_1}, s_{\Pi_2}, \ldots\}$ calculated by the SG, the GG returns a stable s^*_Π, a NE solution that guarantees 1) the plan schedules of all the agents in $\mathcal{AG}$ are executable; and 2) no agent will deviate from its course of action in s^*_Π because no agent can do better by unilaterally changing its strategy. In the case that the schedule profile for every plan profile is unfeasible then the task is *unsolvable*. That is, there is not an executable combination of the agents' strategies.

Since agents operate in a non-strictly competitive environment, the GG is designed as a *general-sum game* or non-zero sum game (Shoham et al. 2009) (Osborne et al. 1994). In this type of games there can be win-win situations because, unlike competitive games, general-sum games feature situations where one decision agent's gain (or loss) does not necessarily result in the other decision agents' loss (or gain).

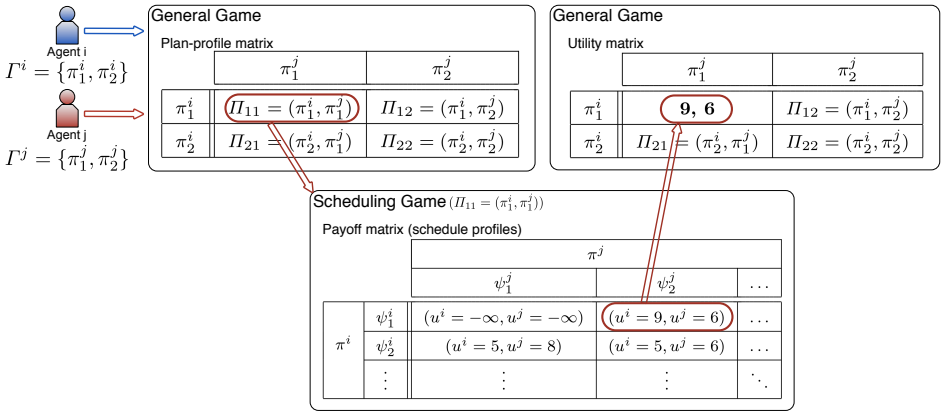


Figure 3.1: An iteration of FENOCOP

Figure 3.1 shows graphically an example of FENOCOP for two self-interested agents i and j, each having two strategies $\Gamma^i = \{\pi_1^i, \pi_2^i\}$ and $\Gamma^j = \{\pi_1^j, \pi_2^j\}$, respectively. The two upper matrices represent the GG in normal or strategic form. This form is given by the two sets Γ^i and Γ^j of agents' strategies (*plan-profile* matrix on the left), and two real-valued utility functions defined on $\Gamma^i \times \Gamma^j$, representing the payoffs to both agents (*utility* matrix on the right). The bottom matrix represents the internal *Scheduling Game*. The SG is actually the game that computes a stable, PO and fair schedule profile s_Π for each plan profile Π. Thus, for each cell in the plan-profile matrix, the GG invokes the SG, which returns the utility received by each agent with s_Π. For instance, in Figure 3.1, the SG is called to compute a feasible schedule profile for Π_{11},

selecting the outcome $s_{\Pi_{11}} = \{\psi_1^i, \psi_2^j\}$, which is then stored in the utility matrix of the GG. Note that the top left cell of the payoff matrix denotes an unfeasible schedule that reports a utility value $-\infty$ to both agents. Once the utility values of all plan profiles are stored in the utility matrix, the GG returns a stable solution s_Π^*.

The key novelty of FENOCOP with respect to other game-theoretic approaches like (Bowling et al. 2003; Larbi et al. 2007) is the introduction of a planning algorithm in the form of a game, the *Scheduling Game*, to compute the payoffs of the plan profiles. Specifically, these two works propose a framework equivalent to our top-level GG, but there is no indication on how to actually achieve a feasible schedule profile that accommodates the plans of all the agents.

3.3 The General Game

The top-level game of FENOCOP, called the General Game (GG), aims to select a stable (NE) schedule profile among the combinations of the agents' strategies. The GG is then modelled as a non-cooperative general-sum game represented in the *normal-form*. This type of game is defined by its *players* (agents), the *strategies* or plans among which they can choose, and the *payoffs* they will each receive for a given strategy. Formally, the GG is defined as follows:

Definition 3.3.1. ***General Game (GG)***. *The GG is a general-sum game with an associated triple* $(\mathcal{AG}, \Gamma, u)$*, where:*

- $\mathcal{AG} = \{1, \ldots, n\}$ *is the set of* n *rational and self-interested agents, the* players *of the GG.*
- $\Gamma = \Gamma^1 \times \ldots \times \Gamma^n$ *represents a finite set of combinations of the agents'* strategies *or plan profiles. A plan profile is a set of plans of the form* $\Pi = (\pi^1, \pi^2, \ldots, \pi^n)$*, where* $\pi^i \in \Gamma^i$ *for each agent* $i \in \mathcal{AG}$*.*

- $u = (u^1, \ldots, u^n)$ *is a set of utility functions, where* $u^i : \Upsilon^i \rightarrow \mathbb{R}$ *is the real-valued* payoff *function for agent* i. $u^i(\psi^i)$ *is the utility that a particular schedule* ψ^i *of plan* π^i *reports to agent* i.

We must note that the payoff that a particular strategy or plan π^i reports to agent i depends on how π^i is combined with the rest of plans of the plan profile Π; i.e., the actual utility is given by the schedule profile $s_\Pi = (\psi^1, \ldots, \psi^i, \ldots, \psi^n)$ returned by the SG. s_Π will determine the specific plan schedule ψ^i for each agent i, which in turn determines the utility obtained by agent i in the plan combination, $u^i(\psi^i)$.

In order to create the utility matrix of the GG, agents launch $\Gamma^1 \times \ldots \times \Gamma^n$ instances of the SG, one per plan profile Π, and the SG computes a schedule profile s_Π along with the utility that s_Π reports to each agent. Once all the agents' utilities are in place, solving the GG means to compute the final solution s^*_Π. This schedule profile constitutes a NE stable solution from which no agent will benefit from invalidating another agent's plan schedule.

The structure of the game, the rationality of the agents and the payoffs that agents' receive are common knowledge in the GG, as it is commonly adopted in game-theoretic approaches in which all information is public.

3.4 The Scheduling Game

As described in Section 3.2, the Scheduling Game (SG) is invoked for each combination of strategies or plan profile $\Pi = (\pi^1, \ldots, \pi^n)$ of the GG in order to retrieve a feasible (executable) schedule profile s_Π that satisfies stability, Pareto optimality and fairness, if such a schedule profile exists. The SG is structured around the following two stages:

1. *Synthesis of schedule profiles.* The SG computes the schedule profiles that coordinate the agents' strategies of the plan profile Π. The resulting payoff matrix (see bottom matrix in Figure 3.1) contains the utilities that the schedule profiles report to each participant.

2. *Schedule profile selection.* Agents solve the game in order to select a stable, PO and fair outcome.

In the first stage of the SG, agents coordinate their plans to guarantee that they are executable in a shared environment. Given a schedule profile s_Π, agents verify that each joint action $A_t \in s_\Pi$ is executable; otherwise, empty actions ($\perp$) are introduced in A_t in order to solve the conflicts that prevent A_t from being executable in a state S. The introduction of an empty action defers the execution of an action of A_t to a later time step $t' > t$. The number of empty actions that an agent i can introduce in a plan schedule $\psi^i \in s_\Pi$ is delimited by λ^i_Π, and hence, there is a finite number of schedule profiles for any given plan profile Π.

After synthesizing the schedule profiles for Π, the self-interested agents jointly select an outcome that maximizes their utilities by taking into account the plan schedules of the other participants. Since a conflict between a subset of plan schedules renders the whole schedule profile unfeasible, every agent i receives a utility $u^i(\psi^i) = -\infty$ for its plan schedule ψ^i in an unfeasible schedule profile. For this reason, we can affirm that the loss of utility of an agent is not the utility gain of the other agents; and so, the SG is a non-strictly competitive problem modelled as a general-sum game. Formally:

Definition 3.4.1. ***Scheduling Game (SG)****. The SG is a general-sum game defined by an associated tuple* $(\Pi, \mathcal{AG}, \Psi_\Pi, u)$*, where:*

- $\Pi = (\pi^1, \ldots, \pi^n)$ *is a combination of plans or* plan profile *for which the SG must find an executable schedule profile* s_Π.

- $\mathcal{AG} = \{1, \ldots, n\}$ *is the set of* n *rational and self-interested agents or* players.

- $\Psi_\Pi = \Psi^1_\Pi \times \ldots \times \Psi^n_\Pi$ *is the set of schedule profiles for the plan profile* $\Pi = (\pi^1, \ldots, \pi^n)$ *represented in the payoff matrix (see Figure 3.1), where each agent* i *has a finite set of* strategies $\Psi^i_\Pi = \{\psi^i_0, \psi^i_1, \ldots, \psi^i_k\}$, *where* $\Psi^i_\Pi \subset \Upsilon^i$, *the possible schedules of its plan* $\pi^i \in \Pi$.

- $u = (u^1, \ldots, u^n)$ *where* $u^i : \Upsilon^i \to \mathbb{R}$ *is a real-valued* payoff function *for agent* i. $u^i(\psi^i)$ *is defined as the utility of the schedule* $\psi^i \in \Psi^i_\Pi$ *when executed in a schedule profile* $s_\Pi = (\psi^1, \ldots, \psi^{i-1}, \psi^i, \psi^{i+1}, \ldots, \psi^n)$. *If* s_Π *is unfeasible, then* $u^i(\psi^i) = -\infty$ *for all agents.*

The set of plan schedules, Ψ^i_Π, that agent i uses to combine its plan $\pi^i \in \Pi$ with the rest of plans of Π is a finite subset of Υ^i. Considering, as stated in Equation 3.1, that the number of empty actions of any plan schedule ψ^i is limited by λ^i_Π, the number of plan schedules in Ψ^i_Π is given by all the combinations that can be formed with the actions in π^i and up to λ^i_Π empty actions.

3.5 Solving the Scheduling Game

This section is devoted to explain three different solving algorithms for the SG. First, we motivate the relevance of three well-known *solution concepts* in non-cooperative game-theory; namely, Nash equilibrium, Pareto Optimality and fairness. Next, in Section 3.5.2, we present two key properties of the SG that will strongly contribute to guarantee the solution concepts of a schedule profile. The following two subsections explain the *normal-form* and *extensive-*

form SG algorithms, respectively. Both algorithms follow the two stages of the SG presented in Section 3.4 and compute solutions that meet the three aforementioned concepts. Additionally, Section 3.5.5 presents an alternative algorithm to obtain a solution to the SG which is a Subgame Perfect Equilibrium (SPE) (Selten 1975). A SPE is also NE but it does not necessarily accomplishes the Pareto optimality and fairness conditions. Finally, last section introduces a problem example of the SG.

3.5.1 Solution Concepts in Non-Cooperative Games

Nash Equilibrium. A Nash Equilibrium (NE) or *stable* solution reflects the best response of an agent taking into account the responses of the rest of agents. In an equilibrium, no agent can benefit from deviating unilaterally from a joint solution. In the SG, a NE outcome is a schedule profile in which an agent cannot improve its utility unless another agent changes its plan schedule. Since a SG can have several NE outcomes (feasible or unfeasible schedule profiles), we introduce a second criterion to choose among them, Pareto optimality.

Pareto optimality. When considering the idea of optimality in multi-agent planing, cooperative planners focus on the optimality of the joint plan or *utilitarian social welfare*, which is typically based on the cost of the actions. In situations where agents are given payments for their participation and optimal solutions are computable, a Vickrey-Clarke-Groves (VCG) (Vickrey 1961; Clarke 1971; Groves 1973; Nissim and Brafman 2013) payment can ensure that the globally optimal plan is also stable. Otherwise, such a globally optimal plan may be very unattractive for some individuals.

Instead of pursuing global optimization, we focus on finding a schedule profile for which we know that there is no other schedule profile that is at least as

good for all agents, and strictly better for one. This best equilibrium schedule profile is called a Pareto Optimal (PO) schedule profile and reflects a situation where no agent can be better off without making at least one agent worse off.

Fairness. Fairness is a criterion that applies to the satisfaction of the agents with their individual utilities. More precisely, an outcome is fair if it maximizes the minimum utility received by any agent; i.e., the least satisfied agent is as satisfied as possible. Basically, the idea is to analyze the schedule profiles in terms of the individual satisfaction of the participants in order to ensure a proper balance of the agents' utilities.

The *egalitarian* principle in ethical theory asserts that all the individuals should enjoy equal benefits from the society (Rawls 1971). As long as there is a positive trade-off between the utility of different individuals, the egalitarian principle leads to the same social choices as the *maxmin* principle, which maximizes the utility of the most unfortunate individuals of a society (*egalitarian social welfare*) (Myerson 1981). In this way, a resource allocation amongst agents in multi-agent systems is considered fair if it is egalitarian (Chevaleyre et al. 2006; Endriss et al. 2006).

In the context of the SG, egalitarian social welfare guarantees that the least satisfied agent has the minimum possible delay. Given a set of NE and PO schedule profiles for a plan profile Π, denoted by $\Omega_\Pi \subseteq \Psi_\Pi$, we define a *fair schedule profile* $\widehat{s_\Pi} \in \Omega_\Pi$ as the schedule profile that results from the application of the max-min utility criterion over Ω_Π:

$$\widehat{s_\Pi} = \arg\max_{s_\Pi \in \Omega_\Pi} \left(\min_{i \in \mathcal{AG}} u^i(s_\Pi) \right) \tag{3.2}$$

The schedule profile that maximizes the utility of the agent which has less utility among the schedule profiles of Ω_Π is selected as the fair solution $\widehat{s_\Pi}$ of

the SG. More than one fair solution can be found if several schedule profiles with the same max-min utility exist in Ω_{Π}.

3.5.2 Properties of the Scheduling Game

The SG features two properties that can be enunciated as follows:

- *Monotonicity*. A SG is said to be *monotonic* if the utility $u^i(\psi^i)$ of any plan schedule $\psi^i \in \Psi^i_{\Pi}$ decreases according to the number of empty actions $\perp$ in ψ^i, and if the schedule profile is feasible. In other words, given two plan schedules ψ^i_x and ψ^i_{x+1}, if ψ^i_{x+1} has more empty actions than ψ^i_x, then $u^i(\psi^i_x) > u^i(\psi^i_{x+1})$. In Definition 3.1.6, we stated that the loss of utility of a plan schedule is only dependent on the finish time of the schedule (except for conflicts). Consequently, every SG is monotonic.
- *Order*. A SG is *ordered* if the strategies of the agents are ordered by decreasing utility in the game. More precisely, if the game is monotonic, for an agent $i \in \mathcal{AG}$, the strategies of Ψ^i_{Π} are ordered from 0 to λ^i_{Π} empty actions.

We note that the utility in the SG is only influenced by conflicts and empty actions. Additionally, one agent does not influence another agent's utility except through the conflicts.

Proposition 3.5.1. *In a SG, if the schedule profile formed by the ideal schedule ψ^i_0 of each agent i is feasible, then this schedule profile is the only outcome of the SG which is both NE and PO.*

Proof. The schedule profile composed of the ideal plan schedules returns the highest utility, $u^i(\psi^i_0)$, for each agent i, and has no empty actions. If such a schedule profile is feasible then this will be a stable and PO outcome because

all agents obtain their highest utility. This schedule profile will also be unique because any other schedule profile will have less utility for at least one agent, since the SG is monotonic. □

Theorem 3.5.1. *In an ordered monotonic SG, any PO schedule profile is a NE.*

Proof. Given that the schedule profiles Ψ_Π of the SG are in decreasing utility order (order property), which decreases with the number of empty actions (monotonicity property), a feasible schedule profile s_Π with maximum utility for an agent i is PO if, for any other feasible schedule profile s'_Π with the maximum utility for the agent i, the utility of the other agents is not higher than their utility in s_Π. In this situation, all the agents in $\mathcal{AG}$ are in best response; and thus, s_Π is a PO NE schedule profile.

By contradiction, suppose a change in strategy of an agent j from a PO profile increases its utility: if j does not reduce its empty actions, but then the utility is not increased, so it must be reducing its empty actions. In this latter case, if a conflict is introduced, its utility is decreased to $-\infty$, so this is a contradiction. If no conflict is introduced, the strategy profile we started with would not be PO to begin with because i's utility is not changed and j's is improved, this is a contradiction again. So j cannot change its strategy to increase its utility, so the PO schedule profile is also a NE. □

In an ordered monotonic SG we only need to seek PO outcomes because a PO outcome s_Π is always a NE, which guarantees that no agent will be willing to deviate from its strategy in s_Π. Therefore, any potential solution of the SG is a PO NE schedule profile. In contrast, in the well-known Prisoner's dilemma, the situation is the opposite, all the outcomes are PO except for a single outcome that is a NE.

In the SG, not every NE schedule profile is necessarily PO and it can actually be an unfeasible outcome. In the example of Table 3.1, the top left cell is a NE with utility $u^i(\psi^i) = -\infty$ for both agents. This happens because there is no better response for those strategies (all the cells that involve the optimal strategy ψ_0^i for any agent i are unfeasible outcomes with $u^i(\psi_0^i) = -\infty$). For this reason, a solution of the SG must not only be a NE, but also PO.

Table 3.1: SG example in normal-form for two agents

	ψ_0^2	ψ_1^2	ψ_2^2
ψ_0^1	**$-\infty$, $-\infty$**	$-\infty$, $-\infty$	$-\infty$, $-\infty$
ψ_1^1	$-\infty$, $-\infty$	$-\infty$, $-\infty$	**9, 8**
ψ_2^1	$-\infty$, $-\infty$	**8, 9**	8, 8

Corollary 3.5.1. *If there is at least one feasible schedule profile for an ordered and monotonic SG, there will be at least a PO NE solution for the game.*

The definition of Pareto optimality establishes that a schedule profile $s_\Pi = (\psi^1, \ldots, \psi^n)$ is PO if it is not Pareto dominated by any other schedule profile $s'_\Pi = (\psi^{1'}, \ldots, \psi^{n'})$; that is, $u^i(\psi^i) \geq u^i(\psi^{i'}),\ \forall i \in \mathcal{AG}$ and $u^i(\psi^i) > u^i(\psi^{i'})$ for some $i \in \mathcal{AG}$. From this definition, it can be drawn that every game must have at least one such optimum (Shoham et al. 2009, Chapter 3). Given that any PO outcome is a NE according to Theorem 3.5.1, if at least one feasible schedule profile exists, there is a PO NE solution for the SG.

3.5.3 Normal-Form SG Algorithm

Given an ordered monotonic SG, the normal-form algorithm obtains all fair PO NE feasible schedule profiles (solutions) of the game. Particularly, this algorithm generates (in the worst-case) all the possible schedule profiles by all the combinations of the schedules of the agents in a decreasing utility order.

In this case, the strategies of each agent are the possible schedules of its plan. The idea is to generate each schedule profile and check its feasibility. Hence, for each schedule profile, agents know their utility and the others' utility since the game structure and rationality of the agents is common knowledge. The algorithm applies a Breadth-First Search (BFS) where each node of the search tree represents a specific schedule profile $s_\Pi = (\psi^1, \ldots, \psi^n)$. The algorithm can be summarized as follows:

1. The root node of the tree is a schedule profile that contains the ideal or highest-utility plan schedule for each agent; i.e., $s_\Pi = (\psi_0^1, \ldots, \psi_0^n)$.

2. The feasibility of a schedule profile is checked at the time of expanding the node. If s_Π results unfeasible, its children nodes are generated. A successor node changes the plan schedule of a single agent in s_Π by its next plan schedule in decreasing order of utility; for instance, the children of $(\psi_0^1, \ldots, \psi_0^n)$ are $(\psi_1^1, \psi_0^2, \ldots, \psi_0^n)$, $(\psi_0^1, \psi_1^2, \ldots, \psi_0^n)$ ... $(\psi_0^1, \psi_0^2, \ldots, \psi_1^n)$. In case that s_Π is feasible, the algorithm applies the PO and fairness conditions over s_Π in order to check whether or not s_Π Pareto dominates and is fairer than any previous feasible node.

3. The search concludes when there are no more nodes to be expanded. At this point, the algorithm returns the set $\widehat{s_\Pi}$, which comprises the nodes of the tree that represent NE, PO and fair solutions.

Figure 3.2 shows an illustrative example of the BFS tree. This example includes three agents (named 1, 2, and 3), each having three different plan schedules (ψ_0^i, ψ_1^i and ψ_2^i, for each agent i). The numbers in squares are the node identifiers and the pa labels indicate the *pivot agent* of the node (see details below).

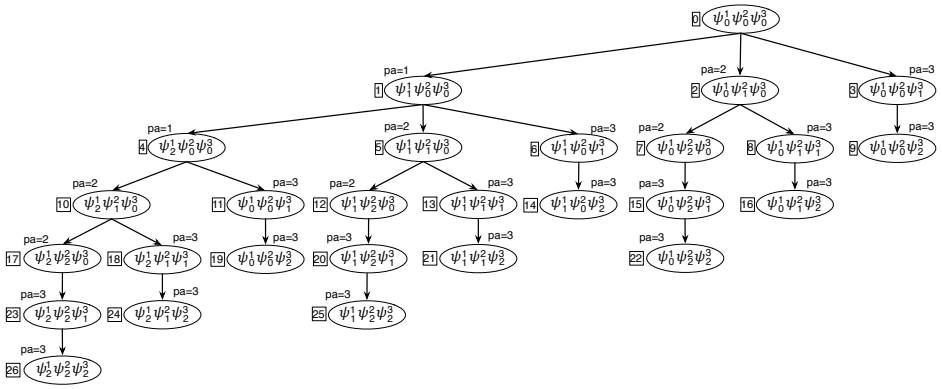


Figure 3.2: BFS tree with all the schedule profiles for 3 agents and 3 schedules per agent

Algorithm 1 details the normal-form SG procedure. The initial schedule profile, consisting of the ideal schedule of each agent, is added to a queue (lines 1-3). The parameter $s_\Pi.pivotAgent$ represents the agent whose plan schedule is changed in s_Π with respect to its parent node. $s_\Pi.pivotAgent$ is used to prevent the generation of repeated or Pareto dominated nodes. The *maxMinBound* parameter stores the maxmin utility of $\widehat{s_\Pi}$ for fairness purposes, and $maxUAg^i$ stores the maximum utility of agent i. Both parameters are initialized to $-\infty$ (lines 4-6).

The *while* loop of the algorithm iterates until the queue of schedule profiles is empty. An iteration of the procedure extracts a schedule profile s_Π from the queue and verifies its *fairness*. s_Π is fair if the minimum utility obtained by an agent in s_Π ($min\, u^j(\psi^j)$, where $\psi^j \in s_\Pi$) is greater or equal than *maxMinBound* (line 9). Otherwise, s_Π is discarded.

Next, the *feasibility* of s_Π is checked by means of the `conflicts`(s_Π) function (line 10). Depending on the result of this verification, different tasks are performed:

- s_Π *is feasible (lines 11-17).* The *Pareto optimality* of s_Π is analyzed by checking that $u^i(s_\Pi) > maxUAg^i$ for at least one agent i in $\mathcal{AG}$ (line 12). If this condition holds, s_Π is confirmed as PO because the agents' schedules are processed in decreasing utility order. Otherwise, s_Π is discarded. If s_Π is fairer than the schedule profiles in $\widehat{s_\Pi}$, (that is, $min\ u^j(s_\Pi) > maxMinBound$), s_Π is stored as the single fair solution in $\widehat{s_\Pi}$. Otherwise, s_Π is added to the $\widehat{s_\Pi}$ set (lines 14-17).

- s_Π *is unfeasible (lines 18-22).* The successor nodes of s_Π are generated and added to the queue. A successor node changes the plan schedule ψ_x^i of an agent i by ψ_{x+1}^i, the next schedule of the agent in decreasing order of utility. The *for* loop (line 19-22) iterates (using the index i) from the pivot agent (stored in $s_\Pi.pivotAgent$) to agent n, generating a total of $n - i + 1$ successor nodes.

The successor nodes of a feasible schedule profile s_Π are not generated because they would be Pareto dominated by s_Π. This conclusion is easily drawn by the monotonicity property, which ensures that the utility of the pivot agent in a successor node is always lower or equal than the utility of its parent schedule profile while the plan schedules of the rest of agents are kept unchanged. This pruning mechanism is correct and it does not remove any solution since, as we showed in Theorem 3.5.1, any PO schedule profile is NE, and hence, there is no need of generating Pareto dominated schedule profiles. All in all, Pareto dominance allows for a meaningful pruning of the BFS search tree.

Complexity of the Normal-Form SG Algorithm

The normal-form algorithm develops a search tree with a maximal branching factor of $|\mathcal{AG}|$. For instance, in the example of Figure 3.2, which includes 3 agents, up to three successors per schedule profile are generated (exclud-

Algorithm 1: Normal-form SG algorithm

```
1  s_Π = ⋃_{i=1}^{n} ψ_0^i;
2  s_Π.pivotAgent = 1;
3  add s_Π to queue;
4  maxMinBound = −∞;
5  for i=1,...,n do
6      maxUAg^i = −∞;
7  while ¬(empty queue) do
8      extract s_Π from queue;
9      if min u^j(ψ^j), ψ^j ∈ s_Π ≥ maxMinBound then
10         if ¬ conflicts(s_Π) then
11             for i=1,...,n do
12                 if u^i(s_Π) > maxUAg^i then
13                     maxUAg^i = u^i(s_Π);
14                     if min u^j(s_Π) > maxMinBound then
15                         maxMinBound = min u^j(s_Π);
16                         ŝ_Π = ∅;
17                     add s_Π to ŝ_Π; break;
18         else
19             for i=s_Π.pivotAgent,...,n do
20                 s'_Π = (ψ^1, ..., ψ^i_{x+1}, ..., ψ^n); ψ^i_x ∈ s_Π; ψ^i_{x+1} ∈ Ψ^i;
21                 s'_Π.pivotAgent = i;
22                 add s'_Π to queue;
23 return ŝ_Π;
```

ing repeated nodes). The maximal depth of the search tree is determined by the sum of schedules of all agents in $\mathcal{AG}$, which is formally defined as $m = \sum_{i \in \mathcal{AG}} |\Psi^i_\Pi|$.

Given the previous considerations, the normal-form SG algorithm presents a worst-case exponential time and space complexity that can be denoted as

$\mathcal{O}(|\mathcal{AG}|^m)$. In practical terms, several mechanisms are applied in order to alleviate the complexity of the algorithm, as illustrated in Figure 3.2:

- The successors of a feasible and PO schedule profile are never generated because the order and monotonicity properties of the SG ensure that all the successors of a feasible PO schedule profile are always Pareto dominated by their parents.

- Cycles in the search tree are controlled in order to prevent the appearance of repeated nodes. For instance, in Figure 3.2, the node $(\psi_1^1, \psi_1^2, \psi_0^3)$ does not appear as a successor of node 2 because it is already included in the subtree of node 1 (see node 5 in Figure 3.2).

- Pareto dominance is also checked among nodes of different subtrees. Let us suppose that the node 1 of Figure 3.2, $(\psi_1^1, \psi_0^2, \psi_0^3)$, is a feasible schedule profile, in which case the subtree of this node would not be generated. The schedule $(\psi_1^1, \psi_1^2, \psi_0^3)$, which is Pareto dominated by node 1, would not either be included in the subtree of node 2, $(\psi_0^1, \psi_1^2, \psi_0^3)$, because the generation of the successors of a node s_Π goes from $s_\Pi.pivot\text{-}Agent$ to n. Since the pivot agent of node 2 is agent 2, its two successors represent a change in the plan schedules of agent 2 and 3, respectively, leaving the schedule of agent 1 unchanged; i.e., ψ_0^1. Consequently, no successor with ψ_1^1 will be generated as a descendent of node 2 even though the subtree of node 1 is not created.

Despite the usage of pruning mechanisms in the BFS tree, the normal-form SG algorithm is a costly procedure that entails exploring most of the schedule profiles in Ψ_Π in order to find a feasible PO and fair solution. Moreover, the branching factor of the search tree is determined by $|\mathcal{AG}|$, which significantly impacts the performance of the algorithm when the number of agents is increased.

3.5.4 Extensive-Form SG Algorithm

In this section, we propose a completely different approach to solve the SG which relies in modeling the problem as an extensive-form tree (Shoham et al. 2009, Chapter 5). The extensive-form algorithm poses the SG as a multi-round sequential game where agents play in turns and incrementally build feasible schedule profiles. This algorithm also obtains all fair PO and NE solutions searching for efficient schedule profiles (Pareto optimality) that present an equitable distribution of the loss of utility caused by the existence of conflicts (fairness).

The extensive-form game is based on a binary tree where agents incrementally generate the schedule profiles for Π action by action. Thus, the branching factor of the tree remains constant regardless of the number of participating agents. This algorithm executes a Depth-First Search (DFS) where a tree node represents the action choice of an agent given the actions introduced in its predecessor nodes.

We note that the extensive-form game is a perfect-information game where agents know the actions and payoff of the other agents. The game structure and agents' rationality is common knowledge. The extensive-form tree can be seen as a simulation of the execution of the plan profile. The execution at time $t+1$ only takes place when every agent has moved at time t, so that players observe the choices of the rest of agents at t. In contrast, the game at time t represents the simultaneous moves of the agents at that time. Simultaneous moves can always be rephrased as sequential moves with imperfect information, in which case agents would likely get 'stuck' if their actions are mutex; that is, agents would not have the possibility of coordinating their actions. Therefore, simultaneous moves at t are also simulated as sequential moves as if agents would know the intention of the other agents. In essence, this can be

interpreted as agents analyzing the possibilities of avoiding the conflict and then playing simultaneously the choice that reports a stable solution. Obviously, this means that agents would know the strategies of the others at time t (the actions applied until t), what seems reasonable if they are all interested in maximizing their utility.

Unlike other games where the agents' strategies are always *applicable*, in planning it may happen that an action a of a plan is not executable at time t in the state resulting from the execution of the $t-1$ previous steps, this is a precondition conflict. In such a case, the schedule profile is discarded. On the other hand, $\perp$ is only applicable at t if at least any other agent applies a non-empty action at t . The empty action is also applicable when the agent has played all the actions of its plan.

Figure 3.3 presents an illustrative example of the tree which includes two different agents, $\mathcal{AG} = \{1, 2\}$. The top left square represents the plan profile of this particular SG, $\Pi = (\pi^1 = [a_1^1, a_2^1], \pi^2 = [a_1^2, a_2^2])$, where preconditions and effects of the actions are shown above and below the nodes, respectively. The nodes of the tree are numbered according to the order in which they are visited by the DFS search. The nodes introduced by agent 1 are depicted in a darker color than those of agent 2. Using this example, we can summarize the behavior of the extensive-form SG algorithm as follows:

1. From the root node, agent 1 generates two successors that represent its possible initial choices, either introducing the first action of its plan, $a_1^1 \in \pi^1$, or an empty action $\perp$ (nodes 1 and 10). At the next level, agent 2 expands node 1 and generates two successors with actions $a_1^2 \in \pi^2$ and $\perp$ (nodes 2 and 6). Next, agent 1 responds by expanding node 2, incorporating actions a_2^1 and $\perp$, respectively. Specifically, the lines labelled as $t = 0$, $t = 1$, etc., delimit the levels of the game; that is, the first game

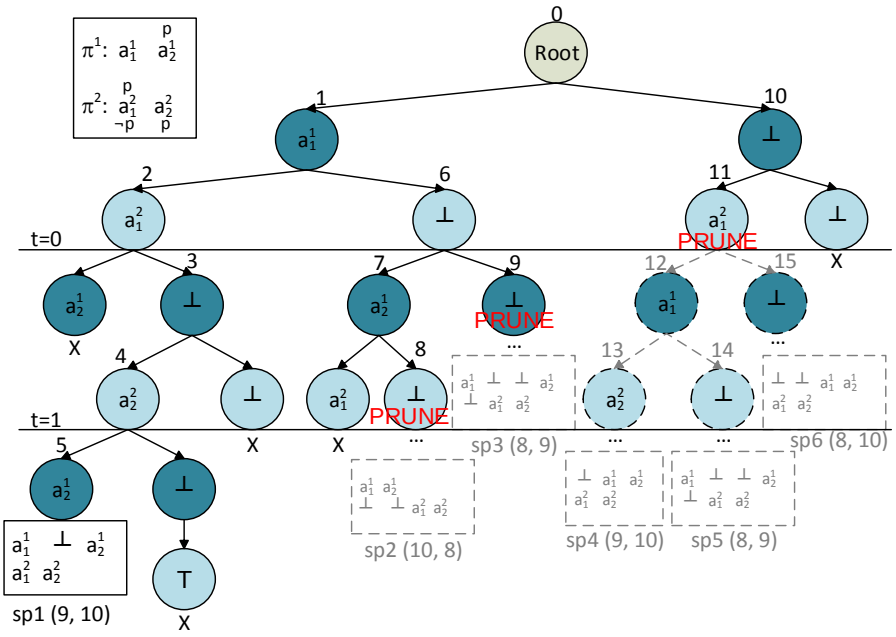


Figure 3.3: SG extensive-form tree example

level comprises the nodes up to $t = 0$, which represent the choices of the joint action A_0; the third and fourth level of the tree represent the second game level ($t = 1$) whose nodes represent the formation of A_1; and so on.

2. For each node, the presence or absence of conflicts is verified to ensure that only feasible schedule profiles are generated. In Figure 3.3, a precondition conflict is detected when agent 1 expands node 2 to insert the action a_2^1 (the precondition $p \in pre(a_2^1)$ does not hold in the corresponding joint state because of the negative effect $\neg p$ of a_1^2). This node is discarded because it does not yield a feasible schedule profile and the algorithm generates the other successor (node 3).

3. Clearly, the intermediate nodes of the tree represent schedule profiles under construction. When a leaf node that contains a fair PO schedule profile is generated, this solution is stored in $\widehat{s_\Pi}$ and it is used as a bound to prune further branches. Given a node nd that represents a partially built schedule profile, we apply an optimistic estimation of the maximum utility that can be obtained from nd by assuming that the expansion of nd up to a solution leaf node does not contain empty actions for any agent. Subsequently, the utility of the estimated solution, say $s_\Pi^{\sim}$, is compared to the utility of the bound. If $s_\Pi^{\sim}$ is unfair or Pareto dominated by the bound, the node nd is pruned. Otherwise, nd is expanded.

   For example, node 5 in Figure 3.3 corresponds to a feasible schedule profile $sp1 \in \widehat{s_\Pi}$ with associated utilities $u^1 = 9$ and $u^2 = 10$. This allows us to prune the following partially built schedule profiles: 1) node 8 because the schedule profile $sp2$ derived from node 8 is unfair compared to $sp1$; 2) node 9 because the resulting schedule profile $sp3$ is Pareto dominated by $sp1$; and 3) node 11 because the expansion of this node would lead to a schedule profile, $sp4$, as good as $sp1$ (the other schedule profiles $sp5$ and $sp6$ are Pareto dominated by $sp1$).

4. The algorithm returns the solutions of the SG when the search is concluded; in our example, $\widehat{s_\Pi} = \{sp1\}$ is the solution found.

The extensive-form algorithm resembles an *alpha-beta* search. On the one hand, a node of the tree represents the move of a player after the moves of its opponents in the preceding levels of the tree. On the other hand, the generation and evaluation of the tree are performed simultaneously and the DFS search ensures that a feasible schedule profile is reached as soon as possible, which will be later used to prune the tree.

The extensive-form algorithm expands first the schedule profiles with fewer empty actions (*monotonicity* property) with the aim to promptly reach a good solution bound. As it occurs in the alpha-beta expansion, the sooner a good bound is reached, the more pruning is applied. On the other hand, note that if the leftmost branch is not pruned, this would represent the ideal schedule of all agents. In short, the DFS expansion together with the chronological backtracking ensures a rational tree expansion, making agents generate first the solutions that report them higher utility (*order* property).

Complexity of the Extensive-Form SG Algorithm

The extensive-form structure is a binary search tree, whose maximal depth is given by the total number of actions of the longest possible schedule profile for the input plan profile Π, which is formally defined as $|s^{-}_{\Pi}| = \sum_{\pi^i \in \Pi} |\pi^i| + \sum_{i \in \mathcal{AG}} |\lambda^i_{\Pi}|$. In other words, each joint action $A_t \in s^{-}_{\Pi}$ includes only one non-empty action for a single agent. We can thus define the complexity of the extensive-form tree algorithm in the worst-case scenario as $\mathcal{O}(2^{|s^{-}_{\Pi}|})$. This is the time and space complexity, since the complete tree could be built and explored in the worst case. We note that the search does not stop when a first solution is found.

However, in practical terms, a substantial part of the tree is pruned in most cases with the best bound found so far and stored in $\widehat{s_{\Pi}}$, thus reducing the overall complexity of the algorithm.

3.5.5 Extensive-Form SPE Algorithm

Similarly to the extensive-form SG presented in Section 3.5.4, this algorithm explores an extensive-form tree via DFS. Instead of ensuring Pareto optimality or fairness, this algorithm computes a Subgame Perfect Equilibrium (SPE). This is also a perfect-information extensive-form game that represents the simultaneous moves of the agents at each time step as sequential moves. Hence, agents are able to avoid conflicts and never get stuck in a conflict by a simultaneous move.

In this algorithm, we apply the SPE (Selten 1975) (Shoham et al. 2009, Chapter 5), which is a solution concept that refines a NE in perfect-information extensive-form games by eliminating those unwanted NE. The SPE of a game are all strategy profiles that are NE for any subgame. By definition, every SPE is also a NE, but not every NE is SPE. The SPE eliminates the so-called "non-credible threats", that is, those situations in which an agent i threatens the other agents to choose a node that is harmful to all of them, with the intention of forcing the other players to change their decisions, thus allowing i to reach a more profitable node. However, this type of threats are non credible because a self-interested agent would not jeopardize its utility.

A standard method to find a SPE in a finite perfect-information extensive-form game is the *backward induction* algorithm (see (Shoham et al. 2009, Chapter 5) for more details and references of this algorithm). This algorithm has the advantage that it can be computed in linear time in the size of the game tree, in contrast to the best-known methods to find NE that require time exponential in the size of the normal-form. In addition, it can be implemented as a single depth-first traversal of the game tree. We consider the SPE as an adequate solution concept for the SG since SPE reflects the strategic behavior of a self-interested agent taking into account the decision of the rest of agents to reach

the most preferable solution in a common environment. However, we must note that in this algorithm, although the solution is a SPE, which is a subset of the NE of the game, it is not necessarily PO and/or fair. These latter properties are not considered in this algorithm since it is an initial version that we used in (Jordán et al. 2015).

The SPE solution concept also has some limitations. First, there could exist multiple SPE in a game, in which case one SPE may be chosen randomly. Second, the order of the agents when building the tree is relevant for the game in some situations. Consider, for instance, the case of a two-agent game. The application of the backward induction algorithm would give some advantage to the first agent in those cases for which there exist two different schedules to avoid a mutex (delaying one agent's action over the other or vice versa). In this case, the first agent will then select the solution that does not delay its conflicting action. Notice that in these situations both solutions are SPE and thus equally good from a game-theoretic perspective. Any other conflict-solving mechanism would also favor one agent over the other one depending on the used criteria; for instance, a planner would favor the agent whose delay returns the shortest makespan solution, and a more social-oriented approach would give an advantage to the agent whose delay minimizes the overall welfare. In order to alleviate the impact of the order of the agents in the SPE solution, agents order is randomly chosen in the tree generation.

We give now a brief explanation of how backward induction will obtain a SPE solution of the SG in the previous tree example of Figure 3.3. If we apply the backward induction algorithm to this extensive-form game, it returns the schedule profile $sp1$, or its equivalent $sp4$. This schedule profile reports the highest possible utility for agent 2, and a penalty of one unit for agent 1. Let's see how the backward induction algorithm obtains the SPE in this example. The payoffs of $sp1$ are back up to node 2, where they will be compared with

the values of node 6. The schedule profile $sp2$ is backed up to node 6 because agent 1 is who chooses which solution goes up between the ones that come from nodes 7 and 9. Then, in node 1 agent 2 chooses between node 2 and node 6 and hence, $sp1$ is chosen. In the other branches, in node 12 $sp4$ will prevail over $sp5$, and then, when compared in node 11 with $sp6$, the choice of agent 1 is $sp4$. Finally, agent 1 chooses at node 0 between $sp1$ and $sp4$, both with the same payoffs, and so both are equivalent SPE solutions. If the tree is developed following a different agent order, the SPE solution will be the same in this particular case.

This algorithm has also pruning mechanisms to avoid generating branches that do not produce SPE solutions. A branch is pruned if the estimation of the maximum utility solution of the branch is strictly lower than any other already computed solution (that would be confirmed as SPE at the end of the process) which act as bounds. In other words, any Pareto dominated solution by an already computed solution is pruned since it would never be a SPE. In the example tree of Figure 3.3, the branches of node 9, node 14, and node 15 are pruned. We note that the pruning annotations of Figure 3.3 do not correspond to the pruning mechanisms that the extensive-form SPE algorithm applies.

Complexity of the Extensive-Form SPE Algorithm

The worst-case time and space complexity of this algorithm is the same as the complexity of the extensive-form SG algorithm, $\mathcal{O}(2^{|s_\Pi^-|})$, where $|s_\Pi^-|$ is the longest possible schedule profile for the input plan profile Π. The pruning mechanisms reduce the overall complexity of the algorithm. However, we note that the pruning of this algorithm is different from the pruning of the extensive-form SG algorithm, and hence, the performance may be also slightly different in practical terms.

3.5.6 Problem Example

We introduce a planning problem of a factory with several depots connected by tunnels. In this problem, the agents are the trucks of the factory, which load and unload their packages in the depots. The tunnels are bidirectional but only one truck at a time can traverse a tunnel. So when a truck enters a tunnel any other truck must wait until the first truck exits. This is a simplification of a real-world problem.

The actions of this domain are: (enter ?truck ?tunnel), (exit ?truck ?tunnel), (load ?truck ?package ?depot), (unload ?truck ?package ?depot). The action enter has a precondition (available ?tunnel) and an effect (not (available ?tunnel)). The exit action has an add effect that restores the availability of the tunnel. Therefore, the actions enter and exit are the only ones that can generate conflicts between the agents.

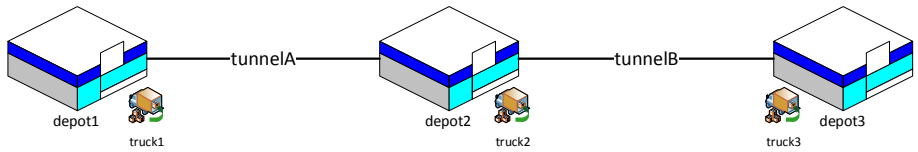


Figure 3.4: Depots and tunnels problem example

Figure 3.4 presents the scenario of the described planning problem where we assume there are three depots, *depot1*, *depot2*, and *depot3*. *depot1* and *depot2* are connected by *tunnelA*; and *tunnelB* connects *depot2* with *depot3*. We have three agents, *truck1*, *truck2* and *truck3*, which are initially located in *depot1*, *depot2* and *depot3*, respectively. There are three packages in the initial state: *package1* is loaded into *truck1*, *package2* is in *depot2* and, *package3* is loaded into *truck3*. Agents have the following plans:

- $\pi^1 = [a_1^1 =$ (enter truck1 tunnelA), $a_2^1 =$ (exit truck1 tunnelA), $a_3^1 =$ (unload truck1 package1 depot2)].
- $\pi^2 = [a_1^2 =$ (load truck2 package2 depot2), $a_2^2 =$ (enter truck2 tunnelA), $a_3^2 =$ (exit truck2 tunnelA), $a_4^2 =$ (unload truck2 package2 depot1)].
- $\pi^3 = [a_1^3 =$ (enter truck3 tunnelB), $a_2^3 =$ (exit truck3 tunnelB), $a_3^3 =$ (enter truck3 tunnelA), $a_4^3 =$ (exit truck3 tunnelA), $a_5^3 =$ (unload truck3 package3 depot1)].

The utility of the earliest schedule for every agent i is $u^i = 10$, and each empty action decreases the utility in one unit. The feasible schedule profiles $sp1$ to $sp4$ represented in Table 3.2 are the only ones which are PO NE (Ω_Π) for this planning problem with utilities: $u(sp1) = (10, 9, 8)$, $u(sp2) = (7, 10, 7)$, $u(sp3) = (10, 7, 10)$, $u(sp4) = (5, 10, 9)$. Table 3.2 shows the joint actions of all the schedule profiles. We use $\top$ to represent that an agent has already executed all the actions of its plan.

The feasible schedule profile $sp1$ is the final fair solution $\widehat{s_\Pi}$ of the SG because it is the only one that accomplishes the three criteria: $sp1$ Pareto dominates any other solution with lower utility for any of the agents (it is also NE in the SG), and it is the only one fair, which maximizes egalitarian social welfare (max-min utility), thus satisfying the most disadvantaged agent. In case of using a global measure like utilitarian social welfare, both $sp1$ and $sp3$ would be valid solutions as the global utility of both is 27. However, agent 2 in $sp3$ is more disadvantaged than in $sp1$, where all agents are more satisfied.

A game-theoretic approach is needed in this context because agents are self-interested and hence, they would deviate from any non-NE schedule profile if they could take profit. In such case, agents might end up in a conflict, thus obtaining a utility of $-\infty$.

Table 3.2: Schedule profiles of the example

sp	ψ^i	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	u^i
$sp1$	ψ^1	a_1^1	a_2^1	a_3^1	$\top$	$\top$	$\top$	$\top$		10
	ψ^2	a_1^2	$\bot$	a_2^2	a_3^2	a_4^2	$\top$	$\top$		9
	ψ^3	a_1^3	a_2^3	$\bot$	$\bot$	a_3^3	a_4^3	a_5^3		8
$sp2$	ψ^1	$\bot$	$\bot$	$\bot$	a_1^1	a_2^1	a_3^1	$\top$	$\top$	7
	ψ^2	a_1^2	a_2^2	a_3^2	a_4^2	$\top$	$\top$	$\top$	$\top$	10
	ψ^3	a_1^3	a_2^3	$\bot$	$\bot$	$\bot$	a_3^3	a_4^3	a_5^3	7
$sp3$	ψ^1	a_1^1	a_2^1	a_3^1	$\top$	$\top$	$\top$	$\top$		10
	ψ^2	a_1^2	$\bot$	$\bot$	$\bot$	a_2^2	a_3^2	a_4^2		7
	ψ^3	a_1^3	a_2^3	a_3^3	a_4^3	a_5^3	$\top$	$\top$		10
$sp4$	ψ^1	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	a_1^1	a_2^1	a_3^1	5
	ψ^2	a_1^2	a_2^2	a_3^2	a_4^2	$\top$	$\top$	$\top$	$\top$	10
	ψ^3	a_1^3	a_2^3	$\bot$	a_3^3	a_4^3	a_5^3	$\top$	$\top$	9

3.6 Conclusions

In this chapter, we presented FENOCOP, a game-theoretic approach for non-cooperative agents that want to execute their plans in a shared environment. Each agent generates a collection of plans that attain its individual task, and takes part on a game that allows the participants to jointly select a feasible schedule profile that guarantees the concurrent execution of their plans. FENOCOP includes two different games: the General Game (GG) is a general-sum game in which agents select the schedule profile to execute among the set of executable combinations of their plans. The generation of a feasible schedule profile for each combination of the agents' plans is taken care of by means of the Scheduling Game (SG). In the SG, agents study how to schedule

their individual plans in order to ensure their executability. Agents address conflicts by delaying the execution of their actions while trying to maximize their utilities.

Whereas the solutions of the GG are guaranteed to be Nash Equilibria (NE), the outcomes of the SG hold two additional solution concepts: *Pareto optimality* allows to return the best outcome among the stable feasible schedule profiles of a SG, and *fairness* maximizes the utility of the least satisfied agent. The satisfaction of these concepts maximizes the quality of the schedule profiles among which the GG selects the solution of the planning problem.

We provide three algorithms to solve the SG problem. The first algorithm represents the SG as a normal-form game to compute the solutions that accomplish the three aforementioned criteria taking advantage of the monotonicity and order properties. Similarly, the second algorithm uses these properties but it builds an extensive-form tree which represents the SG in a different way. The third algorithm is also an extensive-form tree but it computes Subgame Perfect Equilibrium (SPE) which are a subset of the NE of the SG. However, there is no guarantee of Pareto optimality or fairness in this algorithm. In the next chapter, we make an empirical evaluation of these algorithms to analyze the different properties and how they behave experimentally to solve the SG problem. Furthermore, we also evaluate the complete FENOCOP model.

In conclusion, we defined an approach that realistically addresses the non-cooperative multi-agent planning problem. FENOCOP leverages the utilities of self-interested agents and promotes the individual satisfaction of the participants through a set of algorithms which aim for the generation of solutions that are stable, Pareto optimal and fair.

Chapter 4

FENOCOP Experimental Evaluation

This chapter is devoted to experimentally analyze the performance of our FENOCOP (Fair Equilibria in NOn-COoperative Planning) framework. We make a comprehensive empirical evaluation of the Scheduling Game (SG) algorithms in order to assess their performance in different domains. In the case of the extensive-form SPE algorithm, which does not guarantee Pareto optimality and fairness, we analyze the properties of the obtained solutions.

We also evaluate the complete FENOCOP framework, which combines the General Game (GG) and the SG. Particularly, we compare the solutions obtained by FENOCOP against a classical centralized planner. The main goal of this comparison is to prove that the outcomes of FENOCOP are Nash Equilibrium (NE), Pareto Optimal (PO) and fair solutions, since FENOCOP pursues satisfying the

agents' self-interest, in contrast to a centralized planner, which only optimizes a global measure without considering agents' private interests.

The contents of this chapter are organized as follows. Section 4.1 describes the implementation of the FENOCOP framework. In Section 4.2, we introduce the experimental setup of the tests performed in this chapter. Section 4.3 presents a comparative analysis of the three SG algorithms introduced in Section 3.5 of the previous chapter. In Section 4.4, we compare the game-theoretic solutions of FENOCOP with the results of a centralized planner. Finally, in Section 4.5, we discuss the results and draw some conclusions on the experimental analysis.

4.1 FENOCOP Implementation Details

FENOCOP is implemented as a framework that combines the General Game (GG) and the Scheduling Game (SG). Agents in FENOCOP receive a factored description of the problem to solve; that is, each agent receives its own problem file, encoded with the Planning Domain Definition Language (PDDL). Then, each agent i individually computes a set of plans Γ^i via the LPG-td (Gerevini and Serina 2002) planner. These plans constitute the strategies of the GG, and hence, the plan profiles of the GG, which represent the cells of the normal-form GG (see Figure 3.1), are obtained as combinations of the plans of Γ^i for each agent $i \in \mathcal{AG}$.

For each plan profile Π, a SG is solved using one of the algorithms of Section 3.5. Once all the SG instances are solved, the resulting utilities are compiled in the payoff matrix of the normal-form GG. The GG is finally solved by means of the Gambit tool (McKelvey et al. 2014), which computes all the Nash Equilibrium (NE) solutions of the payoff matrix.

The SG algorithms are developed as specified in Section 3.5 using our own software. We implemented and integrated several methods to parse the resulting plans of LPG-td, to check the feasibility of the schedule profiles, to generate the corresponding search trees of the different SG algorithms, to compute the solutions of each SG algorithm, and to process the input and output of Gambit. The only external tool integrated in FENOCOP is Gambit, since it is a robust and fully-tested software to efficiently compute NE solutions.

4.2 Experimental Setup

The benchmarks of this chapter contain problems of two different planning domains:

- *Transport domain.* This domain is inspired by the well-known *zenotravel* domain of the International Planning Competitions (IPC)[1]. Agents are travel agencies that organize their fleets of airplanes to deliver passengers to different destinations. Some of the airplanes are resources shared by the agents; when two or more agents try to use the same plane at the same time, a conflict arises.

- *Space domain.* This is an adaptation of the IPC *rovers* domain. Agents are Mars rovers that navigate through a network of waypoints, analyze samples and communicate the results to a lander, which acts as a communication center. Conflicts arise when agents attempt to analyze the same sample or to simultaneously communicate with the lander.

In these benchmarks, the utility that a plan schedule reports to an agent $i \in \mathcal{AG}$ depends on the *makespan* or finish time of such a schedule. Formally, we define the utility that an agent i obtains from a plan schedule $\psi^i \in \Pi$ as

[1] http://icaps-conference.org/index.php/Main/Competitions

$u^i(\psi^i) = -|\psi^i|$. Therefore, given a schedule profile $s_\Pi = (\psi^1, \ldots, \psi^i, \ldots, \psi^n)$, the utility of an agent i is a value between $-|\pi^i|$ and $-(|\pi^i| + \lambda^i_\Pi)$, depending on the number of empty actions of ψ^i (the schedule of its plan π^i within s_Π).

4.3 Scheduling Game Results

The purpose of this test is to evaluate the performance of the three SG algorithms presented in Section 3.5 of the previous chapter; namely, the *normal-formSG* algorithm (Section 3.5.3), the *ext-formSG* procedure (Section 3.5.4) and the *ext-formSPE* algorithm (Section 3.5.5). Section 4.3.1 presents the performance results regarding coverage, computation time and memory consumption of the SG algorithms. We also show how the *normal-formSG* and *ext-formSG* algorithms achieve fairer results than the *ext-formSPE* version of (Jordán et al. 2015) in Section 4.3.2.

The problems of the benchmark comprise one plan π^i per agent forming a plan profile $\Pi = (\pi^1, \ldots, \pi^i, \ldots, \pi^n)$. We used the planner LPG-td (Gerevini and Serina 2002) to generate the individual plan of each agent in Π. All the problems were run on a single machine[2] with a 30-minute timeout.

The problems of this test feature 2, 3 or 4 agents. The 2-agent and 3-agent problems include from 1 to 6 different resources (planes in the *Transport* domain and samples in the *Space* domain). We created various problem configurations accordingly to the percentage of resources that are shared among the agents (25%, 50%, 75% or 100%). These four variants combined with a maximum number of 6 resources yield 24 different planning settings. For each planning setting, we generated 10 random problems with 2 and 3 agents, totaling 240 problems for each number of agents. In the case of 4-agent problems,

[2] Intel Core i7-3770 CPU at 3.40GHz, 8 GB RAM.

the number of available resources ranges from 1 to 8, which results in 320 problems.

4.3.1 Performance Results

We used three different measures to compare the results of the *normal-formSG*, *ext-formSG*, and *ext-formSPE* algorithms: the *coverage* or number of solved problems, the *computation time*, and the *memory consumption* of each algorithm.

Table 4.1: Problems solved by the *normal-formSG*, the *ext-formSG*, and the *ext-formSPE* algorithms

			normal-formSG			*ext-formSG*			*ext-formSPE*		
$\|\mathcal{AG}\|$	Domain	Solvable	Solved	Partially	Unsolved	Solved	Partially	Unsolved	Solved	Partially	Unsolved
2	*Transport*	164	164 (100%)	0	0	164 (100%)	0	0	164 (100%)	0	0
	Space	240	240 (100%)	0	0	234 (97.5%)	0	6 (2.5%)	226 (94.2%)	8 (3.3%)	6 (2.5%)
3	*Transport*	127	92 (72.4%)	0	35 (27.6%)	127 (100%)	0	0	127 (100%)	0	0
	Space	236	236 (100%)	0	0	216 (91.6%)	19 (8.0%)	1 (0.4%)	212 (89.8%)	23 (9.8%)	1 (0.4%)
4	*Transport*	53	18 (34.0%)	3 (5.6%)	32 (60.4%)	51 (96.0%)	2 (4.0%)	0	51 (96.0%)	2 (4.0%)	0
	Space	297	34 (11.4%)	216 (72.7%)	47 (15.9%)	118 (39.7%)	179 (60.3%)	0	213 (71.7%)	84 (28.3%)	0
Global results		1117	784 (70.2%)	219 (19.6%)	114 (10.2%)	910 (81.5%)	200 (17.9%)	7 (0.6%)	993 (88.9%)	117 (10.5%)	7 (0.6%)

Coverage. Table 4.1 collects the coverage results of the SG algorithms, classified by domain and number of agents. The table shows the percentage of *solved*, *partially solved* and *unsolved* problems for each setting. A problem is said to be *solved* if the search space is exhausted within the 30-minute timeout and so all the problem solutions are found. If the time limit expires before the search space is exhausted and at least one solution is returned, we say the algorithm *partially solves* the problem. Otherwise, if no solution is found, the problem is not solved by the algorithm (*unsolved* problem).

Before running the tests, we executed all the problems without a time limit to verify which problems are *solvable* (see column *Solvable* of Table 4.1). A problem is unsolvable if any of the SG algorithms exhausts the search space

without finding a feasible schedule profile. As shown in Table 4.1, 1117 out of 1600 problems in the benchmark were found to be solvable. Note that the frequent appearance of conflicts in the *Transport* domain leads to more unsolvable problems than in the case of the *Space* domain.

We can observe in Table 4.1 that the algorithms are rather sensitive to the number of agents in the game. The approaches solve most of the problems with 2 and 3 agents, except for the 3-agent *Transport* setting, where *normal-formSG* fails to solve 27% of the problems. However, the performance of the methods clearly degrades with 4 agents: *normal-formSG* solves only 34% of the *Transport* problems and 11% of the *Space* problems, respectively. On the other hand, *ext-formSG* and *ext-formSPE* show a good performance in the *Transport* domain, but 60% of the *Space* problems are only partially solved by *ext-formSG* and 28% in the case of the *ext-formSPE* algorithm.

Table 4.1 also shows that all methods obtain better figures in the *Space* domain. This is explained because in the *Transport* domain a large number of conflicts appear due to the airplanes that are shared by the travel agencies, thus increasing the overall complexity of the game.

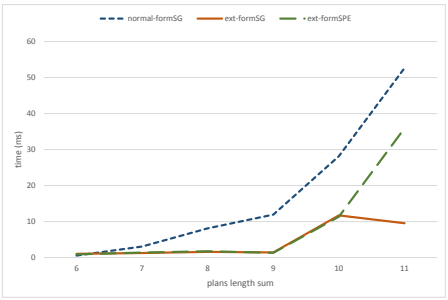
Considering the complexity results of the SG algorithms, clearly the 2-agent setting constitutes the sweet spot of the *normal-formSG* approach, because this algorithm explores a tree which is not as deep as the extensive-form tree. Since the branching factor of the *normal-formSG* method is given by the number of agents, $|\mathcal{AG}|$, its performance significantly degrades in the 3-agent problems and, most notably, with 4 agents, where it is clearly outperformed by the extensive-form approaches. Particularly, it only solves 34% of the *Transport* problems and 11% of the *Space* instances. Nevertheless, *normal-formSG* obtains in general good results, solving almost 70% of the tasks and partially attaining 20% of the instances.

In contrast, the *ext-formSG* and *ext-formSPE* approaches are proven to scale up much better due to the reduced branching factor of the tree and the effectiveness of their pruning mechanisms. In fact, these approaches solve (completely or partially) more than 99% of the benchmark, only failing to generate a solution for 7 problems. Particularly, the *ext-formSG* algorithm solves 81.5% and partially solves 17.9% of the tasks, and the *ext-formSPE* solves 88.9% of the tasks and partially solves 10.5%.

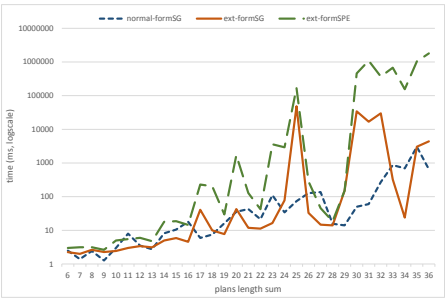
The difference in the coverage results of the approaches is also explained by the fact that the *normal-formSG* does not leverage the appearance of the same conflict in similar schedule profiles. The extensive-form algorithms exploit better this situation because they generate the schedule profiles incrementally. Hence, once a conflict is detected, the unfeasible branch is directly discarded, thus saving a significant amount of computation time.

Computation time. Figure 4.1 compares the computation time required by the SG algorithms. Each data point in the graphs represents the average computation time (in milliseconds) for problems whose input plan profiles have the same number of actions. For instance, a value of 10 on the horizontal axis represents the plan profiles that contain a total of 10 actions. Note that a data point of 1800000 ms in Figure 4.1 (30 minutes) indicates that the timeout was reached in the execution of all the corresponding plan profiles; in other words, these tasks were *partially solved*.

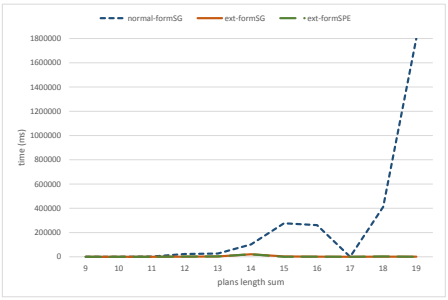
In general, the size of the plan profiles in the *Transport* domain is significantly lower (up to 11, 19 and 27 actions, depending on the number of agents) than the *Space* domain (with a maximum of 36, 54 and 83 actions, respectively). None of the SG algorithms was able to solve the largest instances of the *Transport* domain because of the large amount of conflicts among agents and the size of the problems.



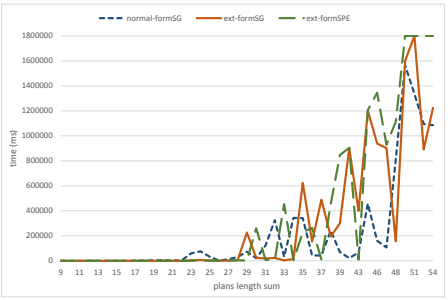
(a) *Transport* domain, 2 agents



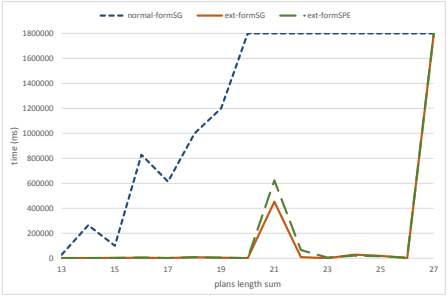
(b) *Space* domain, 2 agents



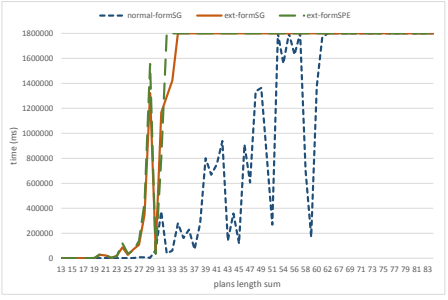
(c) *Transport* domain, 3 agents



(d) *Space* domain, 3 agents



(e) *Transport* domain, 4 agents



(f) *Space* domain, 4 agents

Figure 4.1: Computation time results of the SG algorithms

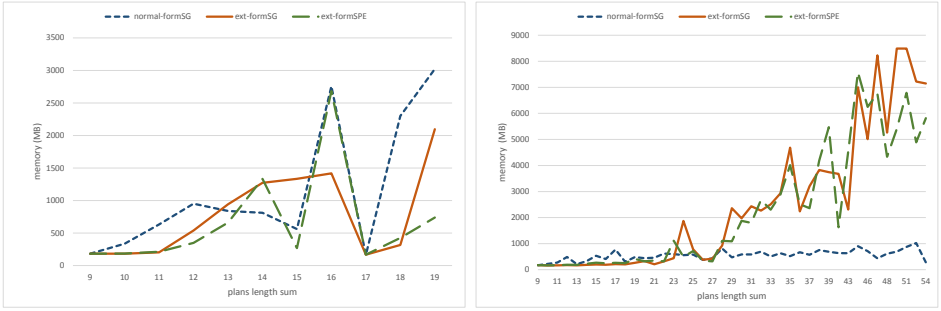
In the *Transport* domain, the extensive-form algorithms clearly outperform the *normal-formSG*, as shown in Figures 4.1(a), 4.1(c) and 4.1(e). This fact becomes more evident in large instances that involve 3 and 4 agents. As noted above, the *normal-formSG* algorithm is particularly sensitive to the number of agents because this parameter determines the branching factor of the search tree. The performance of the *ext-formSG* and *ext-formSPE* algorithms is not so dependent on the number of agents because these algorithms always build a binary tree. *ext-formSG* and *ext-formSPE* solve most of the instances of the *Transport* domain in less than a minute, and scale up significantly better than *normal-formSG*.

Regarding the *Space* domain, the *normal-formSG* approach presents the most stable behavior (see Figures 4.1(b), 4.1(d) and 4.1(f)). In this domain, *ext-formSG* and *ext-formSPE* take a significantly higher amount of time to converge in some problems. The reason is that, in this domain, these algorithms are unable to effectively prune the tree due to the relative lack of conflicts, which results in a large search space to explore. Moreover, *ext-formSG* partially solves a high amount of tasks (particularly in the 4-agent setting), which explains the 30-minute values in Figure 4.1(f). In contrast, the computation times of the *normal-formSG* approach are significantly lower because the order property of the SG is better exploited in this approach. This, together with the low number of conflicts of the *Space* domain, makes the *normal-formSG* a very efficient approach to find feasible schedule profiles in this domain.

All in all, although the *normal-formSG* method slightly outperforms the *ext-formSG* and *ext-formSPE* algorithms in domains that have a low number of conflicts, the extensive-form algorithms present in general a more consistent behavior and they are particularly suitable for problems with a large number of conflicts. The differences between the *ext-formSG* and *ext-formSPE* algorithms are not really significant since both methods are very similar, except

for the pruning mechanisms applied by each algorithm and the additional theoretical properties met by *ext-formSG* (the solutions of *ext-formSPE* are not guaranteed to be Pareto optimal and fair).

Memory consumption. Figure 4.2 shows the memory consumption for 3 agents in *Transport* and *Space* domains. The memory consumption results are less relevant in the *Transport* domain since the amount of problems solved is low. However, the *ext-formSG* algorithm presents a more stable behavior and it consumes less memory than the other approaches in most cases. The pruning applied to unfeasible branches with conflicts reduces significantly the tree expansion of both extensive-form algorithms, which is reflected in the results of the *Transport* domain.



(a) Transport domain

(b) Space domain

Figure 4.2: Memory consumption of *normal-formSG*, *ext-formSG*, and *ext-formSPE* for 3-agent problems

On the other hand, the *normal-formSG* algorithm has an almost constant memory consumption in the *Space* domain as it can be seen in Figure 4.2(b), while the extensive-form algorithms increase their consumption as the problem size grows. The *normal-formSG* approach takes advantage of the order property of the SG and, since the *Space* domain features a low number of conflicts, the

expansion of the *normal-formSG* search tree is lower than the extensive-form trees of the other algorithms. We note that the trees and nodes of *normal-formSG* and extensive-form algorithms are representing different elements. In the first case, each node is a schedule profile which is checked for feasibility, while in the extensive-form trees each node represents an action that is repeated more than once in different branches of the tree. Since the *Space* domain features few conflicts, the pruning techniques applied by the extensive-form algorithms lose effectiveness, which explains why these approaches consume more memory than the *normal-formSG*.

4.3.2 Quality Comparison

In this section, we make a quality comparison between the solutions of the *ext-formSG* approach with respect to the SPE solutions of the *ext-formSPE*. Whereas both approaches yield stable solutions, the *ext-formSG* also guarantees Pareto optimality and fairness. Therefore, the purpose of this test is to assess the quality of the *ext-formSPE* solutions in terms of Pareto optimality and fairness.

Table 4.2 shows the percentage of solutions of the *ext-formSPE* which are PO and fair with respect to the *ext-formSG* (which always yield PO and fair solutions) for the set of solved problems that did not reach the time limit. Since both *normal-formSG* and *ext-formSG* obtain the same solutions, we used the *ext-formSG* approach to perform these tests because it is more efficient. Regarding the *ext-formSPE* approach, since the priority order of the agents is important, it has been chosen randomly in each problem to fairly compare this approach with the *ext-formSG*.

Column *Probs* of Table 4.2 shows the number of solved problems out of the initial set (240 problems for 2-3 agents and 320 for 4 agents) for which none of

Table 4.2: Percentage of PO and fair solutions obtained with the SPE approach

			ext-formSPE	
$\|\mathcal{AG}\|$	Domain	Probs	PO	Fair
2	Tra	164	158 (96.3%)	185 (96.3%)
	Spa	226	221 (97.8%)	193 (85.4%)
3	Tra	127	120 (94.5%)	115 (90.6%)
	Spa	206	205 (99.5%)	124 (60.2%)
4	Tra	51	42 (82.4%)	42 (82.4%)
	Spa	118	118 (100%)	56 (47.5%)
Global results		892	864 (96.9%)	688 (77.1%)

the approaches *ext-formSPE* and *ext-formSG* reached the time limit. Columns *PO* and *Fair* show the percentage of the *ext-formSPE* solutions which are PO and fair with respect to the *ext-formSG* solutions.

In general, most of the SPE solutions are PO with respect to the SG solutions, except for the 4-agent *Transport* problems, where the percentage decreases to 82.4%, as shown in Table 4.2. However, the results of the SPE solutions regarding Pareto optimality can be considered satisfactory for most cases. Globally, 96.9% of the SPE solutions are PO.

As shown in Table 4.2, the percentage of fair solutions of the *ext-formSPE* with respect to the *ext-formSG* is significantly higher for the *Transport* domain than for the *Space* domain. This is because the *Transport* domain has more conflicts and there are fewer possible solutions, and hence, there are many problems in which *ext-formSPE* and *ext-formSG* yield the exact same solution. However, in the *Space* domain, where there are more possible solutions, the percentage of fair solutions of the *ext-formSPE* decreases. Another

interesting result is that the percentage of fair solutions decreases as the number of agents increases. Globally speaking, only 77.1% SPE solutions are fair.

4.4 FENOCOP Results

In this section, we perform an evaluation of the global planning task when agents have several plans in their sets Γ^i, which requires executing both the GG and the SG of FENOCOP. Particularly, we want to compare the quality of the FENOCOP solutions against the solutions of the centralized planner LPG-td (Gerevini and Serina 2002) with respect to Pareto optimality and fairness. We must first point out a couple of observations:

1. The PO and fair schedule profiles returned by the *normal-formSG* or *ext-formSG* algorithms for each plan profile are stored in the utility matrix of the GG (see the illustrative example in Figure 3.1). Then, we solve the normal-form GG and we return a solution that is a NE. However, this does not guarantee that the solution returned by the GG is PO and fair with respect to the other possible outcomes of the GG. Consequently, we must check if the NE solutions returned by the GG satisfy the PO and fairness solution concepts.

2. In LPG-td, we used the standard objective function that minimizes the number of actions so the planner returns the best possible global solution with respect to this optimization criterion. Since a centralized planner does not individually reason on the plan of each agent but on the global plan as a whole, solutions may exhibit a non-equitable distribution of the utilities. This is precisely the key point of comparison between the FENOCOP and LPG-td solutions.

Table 4.3: Solution plans obtained by FENOCOP and the centralized planner LPG-td

	LPG-td					FENOCOP				
	u^1	u^2	u^3	PO	Fair	u^1	u^2	u^3	PO	Fair
tra1	-8	-4	-5	✗	✗	**-5**	**-3**	-5	✓	✓
tra2	-5	-4	-5	✓	✓	-5	-4	-5	✓	✓
tra3	**-3**	-4	-6	✓	✗	-4	-4	**-4**	✓	✓
tra4	-4	**-5**	-4	✓	✓	-4	-6	**-3**	✓	✗
tra5	-4	-4	-3	✓	✓	-4	-4	-3	✓	✓
tra6	-3	-4	-3	✓	✓	-3	-4	-3	✓	✓
tra7	-5	-4	-3	✓	✓	-5	-4	-3	✓	✓
tra8	-3	-3	-5	✗	✗	-3	-3	**-4**	✓	✓
tra9	-3	-5	-5	✗	✗	-3	**-3**	-5	✓	✓
tra10	-5	-3	-6	✗	✗	**-4**	-3	**-5**	✓	✓
tra11	-4	-5	-5	✗	✗	**-3**	**-4**	-5	✓	✓
tra12	-3	-3	-4	✓	✓	-3	-3	-4	✓	✓
tra13	-3	-5	-3	✓	✓	-3	-5	-3	✓	✓
tra14	-4	-3	-4	✓	✓	-4	-3	-4	✓	✓
tra15	-5	-3	-4	✗	✗	**-4**	-3	**-3**	✓	✓
tra16	-4	-4	-4	✗	✗	-4	**-3**	-4	✓	✓
tra17	-4	-4	-4	✗	✗	-4	**-3**	-4	✓	✓
tra18	-10	-6	-12	✗	✗	**-7**	-6	**-4**	✓	✓
tra19	-4	-4	-4	✗	✗	-4	**-3**	-4	✓	✓
tra20	-5	-3	-7	✗	✗	**-4**	-3	**-5**	✓	✓
Global				45%	40%				100%	95%

FENOCOP is the result of integrating a variety of tools, including existing technologies and explicitly designed resources. The input parameters of FENOCOP are the planning tasks of the agents, which are encoded as STRIPS tasks (Fikes et al. 1971). We run LPG-td for solving each planning task and the solution plans are stored in the set Γ^i of each agent. For each combination of plans or plan profile Π, an instance of the SG is launched and solved with one of the

SG algorithms presented in Section 3.5. Once the utilities of the SG solutions are saved in the normal-form matrix of the GG, the NE solutions of the GG are computed by means of the Gambit tool (McKelvey et al. 2014). Finally, from the set of stable solutions, we select a PO and fair outcome (if any) as the final solution of the task.

This benchmark includes 20 3-agent problems from the *Transport* domain. *Transport* was chosen for this test as it is the most challenging domain in our SG benchmark, giving rise to complex instances with a wide variety of conflicts. For FENOCOP, a benchmark problem was encoded as an independent STRIPS task per agent and each task was solved with LPG-td. For running LPG-td as a centralized planner, we encoded each benchmark problem as a single global planning task. We used the standard objective function of minimizing the number of actions in both configurations of LPG-td.

Table 4.3 collects the experimental results of this test. Columns labeled with u^i show the utility of the three agents in both configurations. Note that, as in the SG test, the utility that a plan schedule ψ^i reports to an agent i is defined as $u^i(\psi^i) = -|\psi^i|$. *PO* and *Fair* columns indicate whether or not the plans satisfy Pareto optimality and fairness.

The results clearly prove that FENOCOP is the superior approach when it comes to attain a feasible solution that is also efficient (PO) and represents a balance of the agents satisfaction (equitable loss of utility). As shown in Table 4.3, all the FENOCOP solutions are PO and all but one are fair. Note that the solution of $tra4$ is unfair because the least satisfied agent (agent 2 with $u^2 = -6$) would be more satisfied with the utility of the LPG-td solution ($u^2 = -5$).

As expected, less than half of the solutions of LPG-td when run as a centralized planner are PO and fair. Since the planner decisions are based on global

optimization and non-individualized reason, most of the outcomes are non-efficient and unfair. This is interpreted as a solution that reports a low degree of satisfaction to the involved parties.

Another key advantage of FENOCOP is that, as opposite to centralized planners, we ensure the generation of a stable (Nash Equilibrium) solution. Let us illustrate this through a 2-agent example problem based on the *Transport* domain. The task features two travel agencies (agents) that must organize each a trip for a passenger (p1 and p2, respectively). Passenger p1 starts at city c1 and wants to travel to c2, while p2 is at c2 and wants to visit c4. Aircraft a1 is located at c3 and a2 is at c2.

Table 4.4: Individual agents' plans synthesized by FENOCOP for the example *Transport* task

Time	π_1^1	π_2^1	π_1^2	π_2^2
0	`fly a2 c2 c1`	`fly a1 c3 c2`	`board p2 a2 c2`	`fly a1 c3 c2`
1	`board p1 a2 c1`	`fly a1 c2 c1`	`fly a2 c2 c1`	`board p2 a1 c2`
2	`fly a2 c1 c2`	`board p1 a1 c1`	`fly a2 c1 c4`	`fly a1 c2 c3`
3	`debark p1 a2 c2`	`fly a1 c1 c2`	`debark p2 a2 c4`	`fly a1 c3 c4`
4		`debark p1 a1 c2`		`debark p2 a1 c4`
Utilities	$u^1(\pi_1^1) = -4$	$u^1(\pi_2^1) = -5$	$u^2(\pi_1^2) = -4$	$u^2(\pi_2^2) = -5$

FENOCOP generates two individual plans per agent, as shown in Table 4.4 (π_1^1 and π_2^1 for agent 1, and π_1^2 and π_2^2 for agent 2). The inherent utilities of these plans are also displayed in Table 4.4. The SG is invoked with the four possible combinations of plans ($\Pi_{11} = (\pi_1^1, \pi_1^2)$, $\Pi_{12} = (\pi_1^1, \pi_2^2)$, $\Pi_{21} = (\pi_2^1, \pi_1^2)$ and $\Pi_{22} = (\pi_2^1, \pi_2^2)$) in order to generate four feasible schedule profiles that are stable, PO and fair.

The utilities of the resulting feasible schedule profiles are reflected in the GG utility matrix of Table 4.5, which shows that the solution chosen by FENOCOP is

Table 4.5: GG utility matrix of the example *Transport* task

	π_1^2	π_2^2
π_1^1	-7, -4	-4, -5
π_2^1	**-5, -4**	$-\infty$, $-\infty$

the schedule profile that combines π_2^1 and π_1^2, with associated utilities $u^1 = -5$ and $u^2 = -4$. Despite having the same sum of utilities as the schedule profile that combines π_1^1 and π_2^2 (-9 units), the solution chosen by FENOCOP is the only NE of Table 4.5, since no agent would benefit from unilaterally deviating from this strategy.

We solved this example problem with LPG-td, and the returned solution was π_1^1, π_2^2 (see Table 4.5). As previously stated, a centralized planner, such as LPG-td, optimizes a global metric (in this case, the number of actions of the plan). For this reason, LPG-td considers the solution that combines π_2^1 and π_1^2 as good as the combination of π_2^1 and π_2^2, since they equally minimize the objective function of LPG-td. Hence, LPG-td returns any of these solutions indistinctly. Since the LPG-td solution for this example problem is not a NE, the agents could deviate from this outcome and execute their alternative plans instead, which could potentially cause conflicts that would endanger the executability of the plans.

We can thus conclude that FENOCOP is an appropriate approach to solve the non-cooperative planning problem. Our approach not only guarantees the selection of a Nash Equilibrium solution for the task, but also guarantees that the solutions hold the Pareto optimality and fairness properties in most cases, as empirically demonstrated. In contrast, centralized planners focus on the optimization of global magnitudes, failing in most cases to attain a stable,

PO and fair solution that properly balances the utilities of the self-interested agents.

4.5 General Discussion on the Results

We experimentally validated the algorithms that address the Scheduling Game (SG) problem. We also implemented the overall FENOCOP framework by combining the General Game (GG) together with the SG. The Nash Equilibrium (NE) solution of the GG is obtained by means of the Gambit tool (McKelvey et al. 2014).

The results of the SG algorithms reveal that the *ext-formSG* and *ext-formSPE* approaches scale up significantly better than the *normal-formSG* approach, thanks to their steady branching factor and the effectiveness of their pruning mechanisms. In contrast, the *normal-formSG* algorithm performs slightly better than *ext-formSG* in small instances that involve a reduced amount of conflicts among agents. The solutions obtained by the *ext-formSPE* algorithm are Pareto Optimal (PO) in most cases, but usually are not fair. While the Subgame Perfect Equilibrium (SPE) is generally shown as a good equilibrium concept, it is not enough to guarantee Pareto optimality and fairness as the *normal-formSG* and *ext-formSG* algorithms do. Summarizing, it is recommended to use the *normal-formSG* algorithm in domains that feature few conflicts and the *ext-formSG* when there are more conflicts, while the *ext-formSPE* is not suitable since it does not guarantee PO and fair solutions.

Regarding the FENOCOP results, we confirmed that, as expected, our approach is more effective than a centralized planner at satisfying the agents' interests and fairly balancing their utilities. In contrast, centralized planners focus on

optimizing some metric of the planning task as a whole, being unaware of the presence of self-interested agents with independent objectives.

4.5.1 Limitations of the Model

FENOCOP follows a two-level game scheme which separates the problem of selecting an executable plan combination from the generation of the executable schedule profiles. This division aims to reduce the computational complexity of the task. However, one can observe that, even with this decomposition, the task we aim to solve has exponential complexity, which limits the applicability of FENOCOP to problems of a relatively restrained size.

Furthermore, FENOCOP is limited by the initial set of precomputed plans of each agent, which makes the approach less versatile neglecting the ability of the agents to react to the other agents' plans. From a planning perspective, agents with the capacity to build new plans online would be more complete, thus increasing their problem-solving skills.

For this reason, as a future work, we intend to focus on the synthesis of individual plans. FENOCOP assumes that each agent has a pre-calculated collection of plans and schedules their actions until all the combinations of agents' plans fit together. We believe that the complexity of this costly problem could be significantly alleviated if we equip each agent with the appropriate resources to synthesize a response that directly fits with the rest of agents' plans. Our goal, therefore, is to study the state of the art in multi-agent planning in order to come up with a planning technique that allows an agent to synthesize plans that can be directly integrated with the rest of agents' proposals, thus overcoming the costly task of combining pre-existing individual plans via the delay of individual actions.

This is precisely the purpose of BRPS, the second approach to non-cooperative MAP presented in the context of this PhD thesis, which is introduced and thoroughly analyzed in the next chapter.

Chapter 5

BRPS: Better-Response Planning Strategy for Self-Interested Agents

In some real-life planning problems, agents need to act strategically in order to achieve their goals. Imagine two agents that plan to use a one-capacity resource simultaneously, forcing one of them to wait until the resource is released. Instead, they could come up with a coordinated plan resulting in a better-utility outcome for both and preventing them from conflicts, congestion, and delays.

The non-cooperative MAP problem we aim to solve can be seen as a general-sum game in which agents are self-interested and non-cooperative, but they are willing to collaborate to achieve a stable (equilibrium) conflict-free joint plan that ensures their plans are executable. The problem of finding a plan for

each agent that accommodates their interests into an equilibrium outcome is called in this chapter the *Interaction Planning Game (IPG)*.

In this chapter, we aim to solve the same non-cooperative MAP problem as the one proposed in Chapter 3, but using a different setting. There is a set of rational self-interested agents which want to solve their local planning tasks in a shared environment. However, in this case, agents are able to compute plans during the game, which means they have an unlimited number of plans, in contrast to the model of Chapter 3 where agents have a set of precomputed individual plans. We note that the unlimited set of plans means that agents have planning capabilities to generate different plans and not an infinite set of plans, since this set is limited by the plans cost and the search space of each agent. This feature makes the problem more complete in terms of planning since agents would be able to come up with any plan from their search space that achieves their goals. This is a more realistic setting. We also propose a different formalization of the planning scenario due to the characteristics of this new model.

The exponential complexity and lack of scalability of FENOCOP, the model presented in Chapter 3, motivated the new model of this chapter. In this new model, agents have an unlimited amount of plans (i.e., the ability of building individual plans) and we can take profit from this feature to apply an alternative game-theoretical technique. We use better-response dynamics to obtain only a NE of the problem. Better-response dynamics is a process in which agents iteratively propose new strategies (plans in our case) that improve their utility. This process finishes when no agent is able to propose a new plan that improves its utility, in which case all agents are in their best response, that is, a NE. One of the problems of the former model is the complexity of calculating all NE (usually a subset due to refinements of the approach). We believe that it is not necessary to obtain all NE of our MAP task with self-interested

agents. However, we would not be able to guarantee Pareto optimality or fairness if we do not calculate all NE of a problem. All in all, our aim is to have an efficient approach to tackle with realistic problems, even at the cost of not guaranteeing PO and fair solutions.

Furthermore, we consider congestion interactions, a feature not included in the former model of Chapter 3 which is really important in the game theory literature and representative of real-world problems. Congestion arises when multiple agents use a resource simultaneously thus increasing the cost of using it. A typical example of congestion is traffic flow, where a congested road is more costly to its users (agents) since their pace is slower and the fuel consumption increases.

Finding stable multi-agent plans can be done using the Best-Response Planning (BRP) proposed in (Jonsson et al. 2011). This approach solves *congestion* planning games by applying plan improvements starting with an executable initial joint plan. The best-response dynamics show a remarkable experimental performance but conflicts between the agents' plans are not considered as part of the agents' cost, so convergence and strategic behavior in scenarios that feature planning conflicts are not guaranteed. The theoretical approach in (Jordán et al. 2015) and Chapter 3 presents a combination of two games that computes all the existing equilibria of a joint plan where a conflict between two plans entails $-\infty$ utility for all agents. As we mentioned before, the limitations of this approach are that agents have a limited amount of precomputed plans, congestion situations are not considered and the complexity of the task renders the calculation of all the equilibria intractable.

In an IPG, planning conflicts are considered as part of the agents' costs. This is an important novelty since it allows the agents to reason about them. In better-response dynamics, where each agent proposes its individual plan, it is

necessary to include conflicts as part of the agents' costs to iteratively avoid them, thus achieving a feasible joint plan. In fact, managing conflicts in this way is more accurate since a conflicting situation is not the same with a single conflict, which is nearer to a feasible joint plan, than having multiple conflicts.

The plan of an agent i can provoke a conflict to the plan of another agent j; but, if this conflict does not affect the feasibility of i's plan, an IPG could end up in an equilibrium in which agent j cannot execute its plan. However, in (Boolean) games with a third party that taxes or rewards agents to incentivize them to avoid conflicts, such conflicts can be resolved by a so-called taxation scheme (Nisan and Ronen 2007; Wooldridge et al. 2013). In this work, we show how taxation can also be used for the IPG.

In this chapter, we propose a Better-Response Planning Strategy (BRPS) to solve the IPG. Our approach guarantees convergence to equilibrium joint plans for self-interested planning agents in a setting in which interactions among the agents' plans arise.

Agents in BRPS *play* the IPG using better-response dynamics. Each agent only is able to see the current proposed plans of the other agents, that is, the plans included in the current joint plan. Hence, each agent uses its planning machinery to build a plan that solves its task but considering the interactions with the other agents' plans. Agents only share the public part of their plans, that is, the information that may affect the others. The private information is occluded in the plans they propose in the joint plan. Therefore, while the structure of the game and the existence of the agents is common knowledge, agents ignore the possible strategies of the others as well as their goals.

This chapter is organized as follows. Next section presents the planning problem in which all elements that affect the agents' utility are defined. Section 5.2 formalizes the planning problem as a game-theoretic approach, the

IPG, and we show the complexity of the task as well as under which conditions the IPG is a potential game. In Section 5.3, we present BRPS, the better-response planning strategy to solve the IPG, and we analyze the convergence to equilibrium solutions. Finally, the last section presents the conclusions.

5.1 Planning Scenario

Our planning scenario consists of a set of n individual, rational and self-interested agents, denoted by $\mathcal{AG} = \{1, \ldots, n\}$, where each agent must synthesize a course of action or plan that satisfies its individual goals. Agents design their plans independently of each other and they wish to execute them simultaneously in the same context. Since no agent considers any other's actions when designing its plan, interactions (conflicts and congestions) may arise at the time of executing the plans. In order to avoid this, agents get engaged in a process to coordinate their plans before execution and come up with a joint plan in which every individual plan is executable. This coordination may entail modifications in their originally designed plans. Hence, the objective of an agent is to obtain a minimal-cost plan that satisfies its goals while avoiding interactions, such as conflicts and congestions, with the rest of the agents' plans.

For the sake of clarity, we briefly name all the agents costs which are presented in this section: the cost of an agent plan is $costP$; the cost of solving congestion or conflicts by delaying the execution of plan actions is defined as $costS$; $costG$ represents the cost of being in congestion, and $costU$ is the cost of being in conflict.

Our scenario is modeled as a classical planning problem in a deterministic fully-observable environment. The states of the world are modeled through

a finite set of state variables, $\mathcal{V}$, each of them associated to a finite domain, $\mathcal{D}_v, v \in \mathcal{V}$, of mutually exclusive values that refer to the objects of the world. Assigning a value d to a variable $v \in \mathcal{V}$ generates a *fluent* $\langle v, d \rangle$, indicating that the variable v takes the value d. A state of the world S is a total variable assignment over $\mathcal{V}$ which comprises a fluent for each $v \in \mathcal{V}$. The planning scenario is also defined by a set of actions $\mathcal{A}$ that agents in $\mathcal{AG}$ can execute and a set $\mathcal{R}$ of resources or world objects that are used by the actions in $\mathcal{A}$. Each agent has its own view of the world which may be totally or partially shared with the other agents.

The *planning task* of an agent $i \in \mathcal{AG}$, $\mathcal{T}^i$, is defined as follows:

Definition 5.1.1. *The* **planning task of an agent** $i \in \mathcal{AG}$ *is a tuple* $\mathcal{T}^i = \langle \mathcal{V}^i, \mathcal{I}^i, \mathcal{A}^i, \mathcal{G}^i \rangle$. $\mathcal{V}^i$ *is the set of state variables known to agent* i*. Since agents work in the same context, variables can be shared by two or more agents, i.e.,* $\mathcal{V}^i \cap \mathcal{V}^j \neq \emptyset$. $\mathcal{D}_v^i \subseteq \mathcal{D}_v$ *is the set of values of variable* $v \in \mathcal{V}$ *that are known to agent* i*.* $\mathcal{I}^i$ *is a set of fluents that defines the information of the initial state known to agent* i*.* $\mathcal{I} = \mathcal{I}^1 \cup \ldots \cup \mathcal{I}^n$ *is the initial joint state of all agents in* $\mathcal{AG}$ *with the private information of the agents occluded.* $\mathcal{A}^i \subseteq \mathcal{A}$ *is the set of planning actions or performable operations of agent* i*.* $\mathcal{A}^i$ *and* $\mathcal{A}^j$ *may contain common actions depending on the agents' capabilities.* $\mathcal{R}^i \subseteq \mathcal{R}$ *is the set of resources that agent* i *uses in the preconditions of the actions in* $\mathcal{A}^i$*. A given resource* $r \in \mathcal{R}$ *may be simultaneously used by the actions of two or more agents; i.e.* $\mathcal{R}^i \cap \mathcal{R}^j \neq \emptyset$. $\mathcal{G}^i$ *is the set of goals of agent* i*. Goals of different agents are disjoint sets (*$\mathcal{G}^i \cap \mathcal{G}^j = \emptyset$*) and they must not contradict each other in order to ensure an executable joint plan. An agent is assumed to solve its assigned goals individually, without any assistance or synergy.*

A planning action or performable operation of an agent i is defined as follows:

Definition 5.1.2. *A* **planning action** *of $\mathcal{A}^i$ is a tuple $\alpha^i = \langle pre(\alpha^i), eff(\alpha^i), costA(\alpha^i), resA(\alpha^i)\rangle$, where $pre(\alpha^i)$ and $eff(\alpha^i)$ are partial variable assignments that represent the preconditions (atomic formulae of the form $v = d$) and effects (atomic effects of the form $v := d$) of α^i, respectively; and $costA(\alpha^i)$ is a numeric value that denotes the cost of executing α^i. An action α^i uses a possibly empty set of resources $resA(\alpha^i) \subseteq \mathcal{R}$.*

An action α^i is **executable** *in a state S if $pre(\alpha^i) \subseteq S$. Executing α^i in S leads to a new state S' as a result of applying $eff(\alpha^i)$ over S. An effect $\langle v, d\rangle$ assigns the value d to the variable v, adding the fluent $\langle v, d\rangle$ to S' and removing any fluent from S that contradicts $\langle v, d\rangle$ (i.e., fluents of the form $\langle v, d'\rangle$, where $d' \in \mathcal{D}_v$ and $d' \neq d$).*

Agents develop solutions for their associated tasks in the form of *partial-order plans*. An agent will come up with a partial-order plan that solves its goals which it will attempt to coordinate with the plans of the other agents. The following concepts and definitions are standard notions of the POP paradigm (Penberthy et al. 1992; Ghallab, Nau, et al. 2004), which have been adapted to state variables. Additionally, definitions also account for the local views of the agents' tasks.

Definition 5.1.3. *A* **partial-order plan** *of an agent $i \in \mathcal{AG}$ is a structure $\pi^i = \langle \Delta^i, \prec\rangle$, where $\Delta^i \subseteq \mathcal{A}^i$ is a nonempty subset of the actions of agent i and $\prec$ is a strict partial order on Δ^i.*

Every strict partial order is a directed acyclic graph. Two unordered actions α^i_j and α^i_k of a plan π^i can be executed in any order. Moreover, α^i_j and α^i_k could also be executed in parallel if the agent has the capability to do so. A plan π^i is executable if every topological sort of π^i is executable (Penberthy et al. 1992).

Definition 5.1.4. *A partial-order plan $\pi^i = \langle \Delta^i, \prec \rangle$ is* **executable** *in the initial state of the agent $\mathcal{I}^i$, if for any topological ordering $\langle \alpha_1^i, \ldots, \alpha_m^i \rangle$ of the actions in Δ^i, α_1 is executable in $\mathcal{I}^i$ and any other α_j^i is executable in the state resulting from the execution of the preceding action α_{j-1}^i.*

The set of topological sorts of π^i (linear ordering of a directed acyclic graph) determines a discrete time step for the actions in π^i. Particularly, the time step of an action α^i in π^i is set as the earliest time over every topological sort of π^i. Accordingly, the time step assigned to each action in π^i is consistent with the set of orderings $\prec$ of π^i. The *finish time* of a plan π^i is defined as the last time step t at which any action of π^i is scheduled. The time step of an action α^i in a plan π^i is defined as $time(\alpha^i, \pi^i)$:

$$time(\alpha^i, \pi^i) = \begin{cases} 0, & \nexists \beta^i \prec \alpha^i \\ \max(time(\beta^i, \pi^i)) + 1, & \forall \beta^i \prec \alpha^i \end{cases} \tag{5.1}$$

Figure 5.1 shows π^i, a partial-order plan of an agent i composed of four actions. The time step of action α_1^i is 0, and the time step of actions α_2^i and α_3^i is 1, meaning that the two actions can be concurrently executed by the agent but necessarily after α_1^i because of the ordering constraints. The last action of the plan is α_4^i, which it has to be executed after α_2^i and α_3^i due to the orderings. Once this action is executed, agent i will achieve its goals $\mathcal{G}^i$. Thus, the *finish time* of this plan, i.e., the time at which the goals are achieved, is $t = 2$.

Definition 5.1.5. *A* **solution plan** *for the planning task $\mathcal{T}^i$ of an agent $i \in \mathcal{AG}$ is an executable partial-order plan π^i where all the goals $\mathcal{G}^i$ are achieved.*

The utility of a solution plan to an agent i is measured as the utility that the achievement of $\mathcal{G}^i$ reports to i. Any plan which not achieves all the goals $\mathcal{G}^i$ is not considered. Consequently, two different solution plans π_1^i and π_2^i,

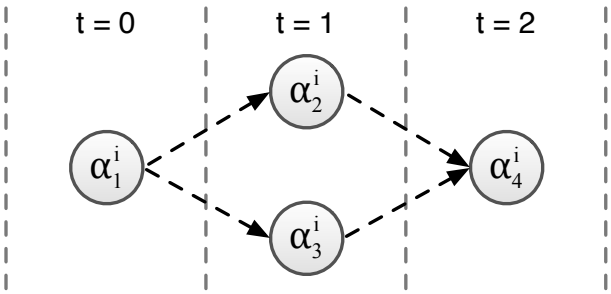


Figure 5.1: An example of partial-order plan for an agent i

that achieve all the goals in $\mathcal{G}^i$, will have the same utility to i. Thus, agents distinguish among solution plans by the cost of performing the plans, denoted as $costP(\pi^i)$. The cost of a plan measures two aspects:

- Cost of the actions in π^i. Each action of the plan is associated to a cost ($costA(\alpha^i)$), which may denote a monetary cost, a cost in terms of resources necessary to carry out the action or any other cost measure that diminishes the benefit of achieving $\mathcal{G}^i$ with π^i. The particular cost of an action to an agent will depend on the context, infrastructure or policy of the agent. For example, the cost of delivering some goods will be higher for agent i than agent j if the vehicle fleet of i is older and less efficient than the fleet of j.

- Finish time of π^i. Time can also play an important role in the cost of a plan. For some agents, achieving the goals sooner or later will have a different impact in the agent's utility. If two plans π_1^i and π_2^i have the same action cost, agent i will most likely prefer the one that finishes earlier. Likewise, the relevance of the finish time of a plan to an agent will depend on the individuality of each agent.

In summary, the term $costP(\pi^i)$ is a value that indicates the effort that agent i must invest to perform π^i. $costP(\pi^i)$ weighs all the relevant parameters to agent i, representing how costly is for agent i to realize plan π^i.

A joint plan is the combination of one plan π^i for each agent $i \in \mathcal{AG}$ in which interactions between the agents may appear:

Definition 5.1.6. *A* **joint plan** *is a tuple* $\Pi = \langle \pi^1, \pi^2, \dots \pi^n, \prec_{\mathcal{AG}} \rangle$ *where* $\prec_{\mathcal{AG}}$ *is a set of inter-agent orderings over the actions of the partial-order plans of the* n *agents.*

For simplicity, we refer to a joint plan as $\Pi = \langle \pi^1, \pi^2, \dots, \pi^n \rangle$, the combination of the individual plans of the n agents. And we use the notation $\Pi^{-i} = \langle \pi^1, \dots, \pi^{i-1}, \pi^{i+1}, \dots, \pi^n \rangle$ to denote the joint plan of all agents but i. Given π^i and Π^{-i}, the aim of agent i is to integrate π^i in Π^{-i} and come up with a joint plan Π.

An *inter-agent partial ordering* is an ordering constraint between two actions α^i and β^j of different agents, i and j, used to address interactions such as conflicts and congestions. The inclusion of an inter-agent ordering of the form $\alpha^i \prec \beta^j$ implies that β^j will be scheduled at least one time step after α^i. This ordering may increase the finish time of the plan π^j of agent j, thus affecting its cost $costP(\pi^j)$.

The time step of an action α in a joint plan Π is defined as shown in the following equation, where the base case of the recursive function is when the time step of all the immediate predecessor actions is 0:

$$time(\alpha^i, \Pi) = \max_{\forall \beta:\ \beta \prec \alpha \ \vee\ \beta \prec_{\mathcal{AG}} \alpha} time(\beta, \Pi) + 1 \tag{5.2}$$

Given a plan π^i and Π^{-i}, the aim of agent i is to integrate π^i in Π^{-i} and come up with a joint plan Π. Ideally, in the absence of any external factors, performing the plan π^i would cost $costP(\pi^i)$ to agent i. However, integrating π^i in Π^{-i} to come up with a joint global plan may incur some additional cost due to the *interactions* (conflicts or congestions) with the other agents. Therefore, agent i examines how costly it is to integrate π^i in Π^{-i} is.

An interaction can be a conflict or a congestion. A conflict may prevent the execution of an action in a plan thus making it unfeasible which is undesirable for agents. A congestion only increases the cost of the implied actions for a simultaneous use of a resource.

5.1.1 Conflict Interactions

In our context, the plans of the agents are executed in the same environment and thereby conflicts between the actions of the agents' plans may arise when building a joint plan Π. A *pair-wise inter-agent planning conflict* or just *conflict* is a situation between two agents in which executing an action of one agent in some specific order may prevent the other agent from performing one of its actions.

In a partial-order plan, a precedence relation $\alpha \prec \beta$ may be imposed by the supporting effect of α ($v := d \in eff(\alpha)$) to the preconditions of β ($v = d \in pre(\beta)$). We will denote such a causal relationship as $\alpha \prec_{\langle v,d \rangle} \beta$.

Definition 5.1.7. *Let π^i, π^j be two plans of agents i and j, respectively, in a joint plan Π. A **conflict** is defined as a tuple $c = \langle \gamma^i, \alpha^j, \beta^j \rangle$ where $\alpha^j \prec_{\langle v,d \rangle} \beta^j \in \pi^j$ and $\gamma^i \in \pi^i$ such that $v := d' \in eff(\gamma^i)$, and it does not hold $\gamma^i \prec_{\mathcal{AG}} \alpha^j$ or $\beta^j \prec_{\mathcal{AG}} \gamma^i$.*

Definition 5.1.7 states a situation in which agent i jeopardizes the execution of π^j (***outgoing conflict*** for i) and, inversely, π^j is affected by agent i (***incoming conflict*** for j). Under a POP paradigm, this interaction is interpreted as the action γ^i is *threatening* the causal link $\alpha^j \prec_{\langle v,d \rangle} \beta^j$; likewise, it amounts to an *inconsistent effect* and an *interference* mutually exclusive relationships (Ghallab, Nau, et al. 2004). That is, in order to avoid this conflict interaction γ^i cannot be executed after α^j and before β^j nor at the same time than α^j or β^j.

Both agents involved in a conflict can adopt the role of conflict solver and attempt solving it by introducing an inter-agent ordering (if possible) which orders their respective actions after the conflicting actions of the other agent. An agent is only allowed to solve a conflict as long as the newly introduced ordering relation is consistent with the rest of orderings of Π ($\prec_{\mathcal{AG}}$) and the plan of the other agent remains unaltered. That is, a conflict is ***solvable*** by any solver agent as long as it does not introduce an incoming inter-agent ordering in the plan of the other agent. This is the key issue that determines the cooperative behavior of self-interested agents; agents seek their own benefit but not at the cost of provoking conflicts to others because this would have a negative impact in all involved agents. Particularly, given a conflict $c = \langle \gamma^i, \alpha^j, \beta^j \rangle$:

- agent i can solve this *outgoing conflict* if it is able to order its action γ^i after β^j; i.e., adding $\beta^j \prec_{\mathcal{AG}} \gamma^i$ to the joint plan.
- agent j can solve this *incoming conflict* if it is able to order its action α^j (and subsequently β^j which goes after α^j) after γ^i by adding $\gamma^i \prec_{\mathcal{AG}} \alpha^j$ to the joint plan.

We must also note that solving a conflict between two agents will not cause any further conflict in the plans of the rest of agents. The theory underpinning the POP paradigm guarantees that new conflicts will only appear in case

new actions are inserted in the agents' plans. However, if there are several conflicts in a plan when integrating it in a joint plan, solving a conflict may provoke another conflict to be unsolvable. Therefore, an agent should analyze the way to minimize the number of conflicts in which it remains.

Integrating π^i in Π^{-i} implies that agent i must successively analyze its incoming and outgoing conflicts with the rest of agents. When an incoming $\prec_{\mathcal{AG}}$ is set to an action γ^i of π^i, the time step of γ^i and its successors must be now calculated with $time(\gamma^i, \Pi)$ over every topological sort that comprises the sets $\prec$ and $\prec_{\mathcal{AG}}$ of π^i. Consequently, the finish time of π^i can be delayed, which has an impact in the integration cost of agent i. Therefore, fixing a *solvable conflict* will impose an *extra cost* to the solver agent due to the possibly delay in the finish time of the plan. The delay cost caused by solving the inter-agent conflicts is included in $costS(\pi^i, \Pi^{-i})$.

From the perspective of an agent i, an ***unsolvable conflict*** is a situation in which agent i is not able to order its action(s) to solve a conflict. This may happen by one of the following reasons:

- Given an outgoing conflict $c = \langle \gamma^i, \alpha^j, \beta^j \rangle$, agent i cannot order γ^i after β^j because it already exists an ordering $\gamma^i \prec_{\mathcal{AG}} \alpha^j$ or $\gamma^i \prec_{\mathcal{AG}} \beta^j$ and hence, adding $\beta^j \prec_{\mathcal{AG}} \gamma^i$ would yield in a cycle of orderings which is not allowed because it is not consistent.

- Given an incoming conflict $c = \langle \gamma^j, \alpha^i, \beta^i \rangle$, agent i is not able to add $\gamma^j \prec_{\mathcal{AG}} \alpha^i$ because it already exists an ordering $\alpha^i \prec_{\mathcal{AG}} \gamma^j$ or $\beta^i \prec_{\mathcal{AG}} \gamma^j$ and this would yield in a joint plan with inconsistent orderings. Similarly, agent i cannot order α^i after γ^j if the action α^i represents one of the values of the initial state $\mathcal{I}^i$.

Our approach also accounts for unsolvable conflicts and charges the agent accordingly in order to encourage the agent to deviate from a conflicting situation and select a strategy that guarantees a feasible joint plan, if possible:

- An ***unsolvable incoming conflict*** $\langle \gamma^j, \alpha^i, \beta^i \rangle$ of agent i compromises the feasibility of π^i and agent i will receive a cost $\boldsymbol{penalty^i}$.
- An ***unsolvable outgoing conflict*** $\langle \gamma^i, \alpha^j, \beta^j \rangle$ of agent i affects the feasibility of π^j. However, the execution of π^i is not compromised in this situation. Since the IPG is a general-sum and non-strictly competitive game, an agent is taxed if its plan provokes an unsolvable conflict. We use a taxation scheme (Nisan and Ronen 2007; Wooldridge et al. 2013) to impose a $\boldsymbol{tax^i}$ to agent i for obstructing the execution of the plan of another agent j.

In order to ensure a conflict-free solution joint plan Π^* (see Definition 5.1.9), the cost of a joint plan with unsolvable conflicts must surpass the cost of any Π^* because it is the worst outcome for any agent. Then, the value of $penalty^i$ and tax^i should be a sufficiently large value that makes π^i be a non-affordable strategy to encourage agent i to deviate from π^i. Since any conflict has the same relevance from the perspective of finding a Π^*, both $penalty^i$ and tax^i are set to a value cc^i that exceeds the cost of the worst possible Π^*. In practice, calculating cc^i is computationally prohibitive so $penalty^i$ and tax^i are assigned a large integer constant *CONF_COST*. Note that cc^i is not set to ∞ because we need to count the number of conflicts to assure convergence to an equilibrium with better-response dynamics, as we will explain in the next sections. Thereby, the cost charged to an agent i for unsolvable conflicts is defined as:

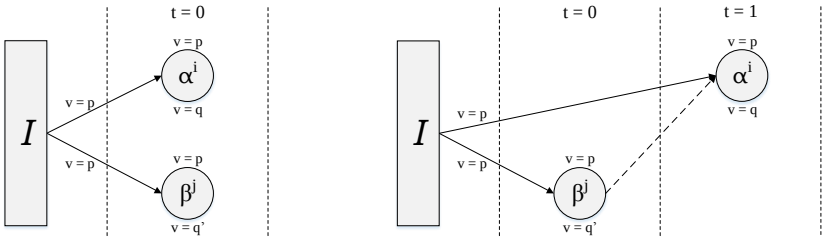
$$costU(\pi^i, \Pi^{-i}) = penalty^i * confIn(\pi^i, \Pi^{-i}) + tax^i * confOut(\pi^i, \Pi^{-i}) =$$
$$= cc^i * (confIn(\pi^i, \Pi^{-i}) + confOut(\pi^i, \Pi^{-i})) \quad (5.3)$$

where $penalty^i$, tax^i, and cc^i are the same constant value *CONF_COST* for agent i, and the functions $confIn(\pi^i, \Pi^{-i})$ and $confOut(\pi^i, \Pi^{-i})$ return the number of incoming and outgoing unsolvable conflicts of plan π^i in the joint plan Π^{-i}, respectively.

$costU$ is a factor which depends on the number of unsolvable conflicts that agent i encounters when integrating π^i into Π^{-i}. In our planning scenario, unsolvable conflicts are translated to cost in order to have a total cost function which determines how good an agent's plan is with respect to the other agent's plans. Particularly, the number of unsolvable conflicts is a key factor since a large number of conflicts denotes a plan that is far away from being executable and so it will be less desirable to the agent. Since the aim of this planning scenario is to consistently reflect the "cost to reach a solution", unsolvable conflicts are also interpreted as a cost factor so as to guarantee convergence to a NE (as we will explain in the next section). This treatment of conflicts is significantly different from the approach (Jordán et al. 2015), where the existence of one conflict is interpreted as $-\infty$ utility regardless of the number of conflicts.

A ***symmetric unsolvable conflict*** is a conflict which is unsolvable by both agent i and agent j, since none of them is able to add an ordering between their affected actions because it will compromise the consistency of the joint plan orderings.

Figure 5.2 shows a complete example of a symmetric unsolvable conflict. As we can observe in Figure 5.2(a), both agents i and j use the same precondition $v = p$ that comes from their respective initial states $\mathcal{I}^i$ and $\mathcal{I}^j$, which can be



(a) Two unsolved conflicts

(b) One symmetric unsolvable conflict

Figure 5.2: Symmetric unsolvable conflict example

seen as an initial joint state $\mathcal{I} = \mathcal{I}^i \cup \mathcal{I}^j$. Both agents modify the value of the variable v as an effect of their respective actions α^i and β^j. This provokes two conflicts: $c_1 = \langle \beta^j, \mathcal{I}^i, \alpha^i \rangle$ and $c_2 = \langle \alpha^i, \mathcal{I}^j, \beta^j \rangle$. In c_1, β^j endangers the execution of α^i because its effect changes the value of v, which is used by α^i at time step $t = 0$. Similarly, the conflict c_2 makes the execution of β^j not feasible. Therefore, from the perspective of agent i, the conflict c_2 is solvable adding $\beta^j \prec_{\mathcal{AG}} \alpha^i$, which allows agent j to execute β^j because the precondition $v = d$ is not in danger. Now the situation is represented in Figure 5.2(b) with the conflict c_2 solved by agent i. However, c_1 is an unsolvable incoming conflict for agent i since it is not possible for i to add the ordering $\beta^j \prec_{\mathcal{AG}} \mathcal{I}^i$ because the initial state $\mathcal{I}^i$ cannot be ordered after any action. From the perspective of agent j, c_1 is an unsolvable outgoing conflict because it is not allowed to introduce $\alpha^i \prec_{\mathcal{AG}} \beta^j$ since it would provoke a cycle thus violating the consistency of the orderings in the joint plan. Thereby, the situation represented in Figure 5.2(b) is a symmetric unsolvable conflict because none of the agents can add an ordering to solve it.

Summarizing, we refer to the different types of conflicting situations that can arise as follows:

- ***Solvable conflict***. An agent i is able to order its affected action(s) after the other agent's action(s) to solve the conflict. The possible cost rise by the delay of the execution of the actions of i is reflected in $costS(\pi^i, \Pi^{-i})$.
- ***Unsolvable conflict***. It is a conflict that cannot be solved by agent i because it is not able to order its affected action(s) after the other agent's action(s) because the orderings of the joint plan would be inconsistent. An incoming or outgoing unsolvable conflict implies a cost which is considered in $costU(\pi^i, \Pi^{-i})$.
- ***Symmetric unsolvable conflict***. It is a conflict in which none of the two involved agents is able to solve it. A conflict like this is not solvable and hence, one of the agents should change its plan in order to find a conflict-free joint plan.

5.1.2 Congestion Interactions

On integrating π^i in Π^{-i}, it may exist actions of different agents that use one resource r at the same time t. This is called *congestion*.

A *congestion game* is defined by players and resources, and the utility of the player depends on the resources and the number of players choosing the same resource (Rosenthal 1973). In our case, certain items in $\mathcal{V}$ are defined as resources or congestible elements ($\mathcal{R}$) so that a congestion is produced when two or more actions associated to the same time step define a formula $v = d, v \in \mathcal{R}$ in their preconditions. Moreover, the cost of a congestion may differ across the agents involved in it, which makes our approach more realistic.

Definition 5.1.8. *A* **congestion** *arises when one or more actions of a plan π^i being integrated in a joint plan Π^{-i} are simultaneously executed at time t with actions of the other agents' plans in Π^{-i} that use one same resource $r \in \mathcal{R}$.*

The simultaneous usage of a resource r is related to a particular time t. The discrete time model in our scenario (as defined in Equation 5.2) is represented with the set $\mathbb{N}$ of natural numbers. We define the function $\# : \mathcal{PL} \times \mathbb{N} \times \mathcal{R} \rightarrow \mathbb{N}$, where $\mathcal{PL}$ is the set of all possible joint plans, as a counting function that returns the total number of actions that use one resource at a time in a joint plan. Specifically, given a joint plan Π, $\#(\Pi, t, r)$ returns the number of actions that use the resource r at time step t:

$$\#(\Pi, t, r) = \sum_{j=1}^{n} \sum_{k=1}^{m} \alpha_k^j \in \pi^j : time(\alpha_k^j, \Pi) = t \ \wedge \ resA(\alpha_k^j) = r \tag{5.4}$$

A congestion among several actions at a time means a more costly realization of these actions, either monetary or because of a later finish time of the plan if the agent avoids the congestion by deferring its congested action(s). Given an action α^i such that $resA(\alpha^i) = r$ and $time(\alpha^i, \Pi) = t$, the original cost value of α^i, $costA(\alpha^i)$, will be increased by a factor that depends on the value $\#(\Pi, t, r)$ and how this increment impacts the cost of agent i. We define $\mathcal{C}_r^i : \mathbb{N} \rightarrow \mathbb{R}$ as the cost function of resource r for agent i accordingly to the number of times that r is simultaneously used. Therefore, the congestion cost incurred by agent i when integrating its plan π^i in Π^{-i} is defined as:

$$costG(\pi^i, \Pi^{-i}) = \sum_{t=0}^{finish(\Pi)} \sum_{r \in \mathcal{R}} \mathcal{C}_r^i(\#(\Pi, t, r)). \tag{5.5}$$

Similar to conflicts, a congestion can be addressed with inter-agent orderings. Given an action α^i scheduled at time t that uses resource r, the congestion is avoidable by agent i by setting a precedence relation $\lambda \prec_{\mathcal{AG}} \alpha^i$ with all the actions λ in congestion with α^i. Addressing a congestion of an agent's plan with inter-agent orderings may imply that the finish time of the plan is modified, thus yielding in a cost rise for the agent. The possible delay cost

caused by this relation in the finish time of π^i is accumulated in $costS(\pi^i, \Pi^{-i})$ as well as the solvable conflicts. On the other hand, if the agent remains on congestion it also implies a cost rise in the function $costG(\pi^i, \Pi^{-i})$. Therefore, depending on the cost impact for the agent it would prefer to address the congestion or remain on it.

5.1.3 Cost of Integrating a Plan in a Joint Plan

As explained above, solving a conflict or addressing a congestion with inter-agent orderings will most likely impose an extra cost to the agent whose plan π^i is being integrated in Π^{-i}. In both cases, the cost rise for the agent i is reflected by the function $costS(\pi^i, \Pi^{-i})$. The cost of remaining on congestion is measured by the function $costG(\pi^i, \Pi^{-i})$. And $costU(\pi^i, \Pi^{-i})$ gathers the cost for the unsolvable conflicts. Therefore, the total cost of agent i when integrating a plan π^i in a joint plan Π^{-i} is defined as the cost of the plan itself $costP(\pi^i)$ and all the costs provoked by solved and unsolved conflicts and congestions:

$$costTotal(\pi^i, \Pi^{-i}) = costP(\pi^i) + \\ + costS(\pi^i, \Pi^{-i}) + costG(\pi^i, \Pi^{-i}) + costU(\pi^i, \Pi^{-i}) \quad (5.6)$$

The function $costTotal(\pi^i, \Pi^{-i})$ returns a value which represents how bad or costly the plan π^i is to agent i under the context of Π^{-i}. The utility of a plan can be measured as the opposite of its cost, as it is commonly adopted in game-theoretic approaches. In our case, the net utility u^i that a plan π^i reports to agent i will be the utility of achieving $\mathcal{G}^i$ minus $costTotal(\pi^i, \Pi^{-i})$.

Finally, a conflict-free joint plan is considered a solution joint plan for the planning scenario of the tasks of the agents.

Definition 5.1.9. *A* **solution joint plan** *for the planning tasks* $\bigcup_{i \in \mathcal{AG}} \mathcal{T}^i$ *of all agents in* $\mathcal{AG}$ *is a* conflict-free *joint plan* Π^* *where* $costU(\pi^i, \Pi^{-i}) = 0, \forall i \in \mathcal{AG}$. *If this condition holds then it is guaranteed that* Π^* *achieves* $\bigcup_{i \in \mathcal{AG}} \mathcal{G}^i$.

5.2 Interaction Planning Game

An ***Interaction Planning Game (IPG)*** is a general-sum or non-zero-sum game to solve the planning scenario presented in Section 5.1. A general-sum game is a game in which agents' aggregate gains and losses can be less than or more than zero, meaning that agents do not try to minimize the others' utilities (Gillies 1959; Shoham et al. 2009, Chapter 3). In an IPG, agents are self-interested but not strictly competitive so the aim of an agent is to seek an individual plan that does not provoke any conflict that would negatively affect its utility as well as the others' utilities. Specifically, a conflict between two or more plans will render the plans non-executable, which is the worst possible outcome for the agents because it prevents them from fulfilling their planning tasks. In this section, we present the elements that define an IPG.

Each agent i has a task $\mathcal{T}^i$ to solve. In other words, an agent i has to generate a plan π^i with its actions $\mathcal{A}^i$ to solve all its goals in $\mathcal{G}^i$. An IPG is defined as follows:

Definition 5.2.1. *An* **Interaction Planning Game (IPG)** *is a tuple* $\langle \mathcal{AG}, \mathcal{T}, u \rangle$, *where:*

- $\mathcal{AG} = \{1, \ldots, n\}$ *is a set of* n *rational self-interested planning agents.*
- $\mathcal{T} = \bigcup_{i \in \mathcal{AG}} \mathcal{T}^i$ *is a multi-agent planning task in which each agent* i *has to solve its own task* $\mathcal{T}^i$.

- $u = (u^1, \ldots, u^n)$ *where* $u^i : \pi^i, \Pi \to \mathbb{R}$ *is a real-valued payoff function for agent* i *defined as the utility of a plan* π^i *that solves task* $\mathcal{T}^i$ *when it is integrated in a joint plan* $\Pi = \langle \pi^1, \ldots, \pi^{i-1}, \pi^i, \pi^{i+1}, \ldots, \pi^n \rangle$.

The strategies of an agent i in the IPG are the possible plans that it can build to solve its own task $\mathcal{T}^i$. We note that agents only reveal their plans when playing their strategies (plans). Additionally, each agent only shows the parts of the plan that are public since they may affect to other agents, while it occludes its private information when sharing the plan with other agents (we give further details about how to solve the IPG using better-response dynamics in next section). Therefore, the game structure and the other players' existence is common knowledge, but the rationality and particular utility of each agent are private information.

An IPG solution must be a joint plan such that the individual solution of each agent within the joint plan cannot be improved; otherwise, agents would keep on altering the "solution", thus leading to instabilities and conflicts during the plan execution. Our goal by modeling this as a game is to guarantee a stable solution in which no agent has a reason to change its strategy. Then, the aim of each agent in the IPG is to select its best-utility strategy according to the strategies selected by the others; that is, all agents must be in best response in an IPG solution, which by definition is a NE (see (Shoham et al. 2009, Chapter 3) for more information and examples).

Definition 5.2.2. *An* **IPG solution** *is a solution joint plan* Π^* *(as specified in Definition 5.1.9) which is a NE of the IPG. An IPG solution is a conflict-free joint plan in which the plan of each agent is in best response with respect to the other agents' plans.*

The complexity of finding a NE in the IPG is PPAD-hard (Polynomial Parity Arguments on Directed graphs). This complexity class was introduced by Christos Papadimitriou in (Papadimitriou 1994). A solution to a PPAD problem is known to exists, but it may be hard to find it. Moreover, Chen and Deng shown in (Chen and Deng 2006) that computing a NE, even in a 2-player game, is PPAD-complete unless $P = NP$. However, there are some exceptions in which for some restricted games, such as zero-sum games, a NE can be computed in polynomial time using linear programming (Shoham et al. 2009, Chapter 4).

Theorem 5.2.1. *Computing a NE for an IPG is PPAD-hard even for single-action plans.*

Proof. This proof uses a reduction from general-sum finite games. It is known that finding a sample NE for these games with two or more players is PPAD-complete (Shoham et al. 2009, Chapter 4). For this class of games, any strategy of any player/agent i can be translated to a task $\mathcal{T}^i$ of the IPG. This is done in polynomial time since a direct mapping of the strategies of any general-sum game can be done to single-action plans of the IPG.

Each of these plans, when combined with the other agents' plans, forms a joint plan which has the utility u^i for each agent i of the corresponding outcome of the original general-sum game, so we will have the set of all joint plans that solve the multi-agent planning task $\mathcal{T}$ of the IPG.

We have equivalent games since any general-sum game can be mapped to an IPG because, in fact, it is a general-sum game. Now, a NE of the IPG can be translated in polynomial time to a NE of the equivalent general-sum finite game, since the strategies and outcomes are the same.

□

This proof is based on the premise that if a NE of the IPG could be computed in polynomial time, then a NE of any general-sum finite game would be computed in polynomial time and thus, the problem of computing a NE would be in P. This is a contradiction because it is already known that computing a NE for a general-sum game with two or more players is PPAD-complete (Chen and Deng 2006), unless $P = NP$ which is unknown but unlikely. From this we can conclude that even if generating plans for individual agents is easy (single-action plans), finding a stable solution is PPAD-hard.

We analyzed the complexity of the IPG from the game-theoretic point of view. Now, we analyze the complexity from a planning perspective since the agents that participate in the IPG must have the ability of building plans and schedule these plans in their best possible way to obtain the best cost.

In the general case, propositional STRIPS planning is PSPACE-complete (Bylander 1994), as well as *SAS+* (Bäckström and Nebel 1995), for both satisficing and optimal planning. Therefore, there is no polynomial algorithm for classical planning unless $P = PSPACE$, and planning is not polynomially reducible to any problem in NP unless $NP = PSPACE$. We note that there is no formal proof for the equalities $P = PSPACE$ and $NP = PSPACE$, but the general belief is that they are false. Nevertheless, planning complexity varies depending on the planning domain. For instance, Helmert 2003 proved that a non-optimal plan can be obtained in polynomial time for a transport domain without fuel restrictions, but bounded (length) plan existence is always NP-complete.

Cost-optimal planning has proven more difficult to solve in practice than propositional STRIPS planning. The existence of zero-cost actions in the domains may result in long plans which are hard to find (Richter et al. 2010; Benton, Talamadupula, et al. 2010). In addition, the same problem happens if

action costs have big differences even without zero-cost actions (Wilt et al. 2011). Cost-optimal planning remains W[2]-hard for all domains (Aghighi et al. 2016). Using parametrized complexity analysis, propositional STRIPS planning is also W[2]-complete when parametrized with plan length (Bäckström, Jonsson, et al. 2015).

Theorem 5.2.2. *The IPG is PSPACE-hard even with just one agent.*

Proof. In this proof, we make a reduction from single-agent planning to an IPG. Let us take any single-agent planning problem which can be represented as a planning task $\mathcal{T}^i$ of an agent i. We can construct an instance of the IPG for agent i with its task $\mathcal{T}^i$ and $\mathcal{AG} = \{i\}$, this step is directly applicable, so it can be done in polynomial time. Then, solving this IPG is only about computing single-agent plans that solve $\mathcal{T}^i$. The IPG solution obtained from this IPG is a plan with a specific utility u^i, which can be translated directly to a plan that is a solution to the initial single-agent planning task $\mathcal{T}^i$. The later translation is made in polynomial time since it is a direct mapping of the plan as it is obtained in the IPG. □

Since it is known that planning is PSPACE-complete in the general case, we can ensure that planning for an IPG is PSPACE-hard.

We have defined all the elements to calculate the value of the function $costTotal$, which determines the utility of each agent in any situation. Now, we introduce the class of potential games and we analyze under which conditions the IPG is a potential game.

As we mentioned before, Rosenthal 1973 presented a class of games called *congestion games*. Monderer et al. 1996 found a more general class named *potential games*. A game is potential if there is a function on the strategies of players such that each change in a player's strategy changes the function

value in the same way as the player's change in utility. For such a potential function, each local optimum is a Pure strategy Nash Equilibrium (PNE) (Voorneveld et al. 1999). This class includes some well-studied network-based games. Some interesting properties of potential games are the existence of pure equilibria, the guarantee of convergence with best-response dynamics, and bounded price of anarchy with the *potential function method* (Roughgarden 2005). In contrast to an exact potential function, an ordinal potential function does not track the exact change of utility of the players but it tracks the direction of such change.

For the IPG, we define the following *ordinal potential function* which maps every strategy profile or joint plan to a real value:

$$\Phi(\Pi) = \sum_{i \in \mathcal{AG}} costTotal(\pi^i, \Pi) \tag{5.7}$$

Any unsolvable conflict causes a huge cost increase cc to the involved agents (a penalty to the affected agent, and a tax to the provoking agent). Since this cost increase is the constant value $CONF\_COST$, which is higher than the cost of any conflict-free plan, it is straightforward to see that agents will always avoid unsolvable conflicts if they can do so. No agent can benefit from being in an unsolvable conflict or provoking it to improve its individual cost, no matter their individual cost functions. In other words, a conflict increases the cost of the involved agents as well as the potential function Φ. Therefore, regarding unsolvable conflicts and how they are taxed in the IPG, the potential game property always holds.

Usually, congestion games have a universal cost function which expresses the congestion caused by the use of the resources of the game. These games are potential if congestion affects all agents similarly. When agents have in-

dividual payoff functions, a game is not potential anymore as it is proven in (Milchtaich 1996). Since switching strategies usually means a change in plan costs, it may be profitable for an agent to change its plan to a much cheaper one that introduces more congestion to others. Under these conditions, the potential game property cannot hold because the potential function is unable to track the improvement of the agent if the losses of the other agents are not compensated. Agents in the IPG have individual costs that affect them differently for their plans ($costP$), for solving congestion or conflicts ($costS$), and for congestion ($costG$).

However, the IPG is a potential game if one of these two sufficient conditions are accomplished: (a) congestion is costless, or (b) agents plans cost are null and congestion affects all agents similarly.

Theorem 5.2.3. *The IPG is a potential game with its associated ordinal potential function Φ if for all agents in $\mathcal{AG}$:*

(a) congestion is costless ($costG = 0$), or

(b) the cost of executing a plan is null ($costP = 0$) and congestion affects all agents similarly.

Proof. The ordinal potential function Φ maps every strategy profile to a real value and it satisfies the following potential game property: Given a joint plan $\Pi = \langle \pi^1, \ldots, \pi^i_x, \ldots, \pi^n \rangle$, if and only if π^i_y is an alternate plan/strategy for agent i, and $\Pi' = \langle \pi^1, \ldots, \pi^i_y, \ldots, \pi^n \rangle \neq \Pi$, then $\Phi(\Pi) - \Phi(\Pi') > 0$ and $u^i(\pi^i_y, \Pi') - u^i(\pi^i_x, \Pi) > 0$. In other words, if the current state of the game is Π, and an agent i switches its strategy from π^i_x to π^i_y, the improvement of i is tracked by Φ.

Regarding congestion, in the case (a) in which congestion is not considered, it is straightforward to see that any utility improvement of an agent by switching

its plan will be reflected in the potential function Φ and it would not cause any cost increase to other agents. In the case (b), congestion affects all agents similarly and the cost of executing any individual plan is null. Hence, an agent incurring in a congestion is as much affected as the other involved agents, and similarly, if an agent avoids a congestion, the other involved agents also increase their utility. Therefore, the potential game property holds in both cases (a) and (b) regarding congestion.

Unsolvable conflicts imply a cost increase of cc to the involved agents, which is higher than any conflict-free plan cost. If an agent i improves its utility by avoiding a conflict, then the potential function Φ will decrease $2cc$, once for each of both agents involved in the avoided conflict. Note that any modification of a plan (increase in $costS$ by solving a conflict) or switching to another plan to avoid a conflict always implies a cost decrease for the involved agents which is tracked by Φ. Hence, the potential game property always holds regarding conflicts in both cases (a) and (b).

□

For potential games, convergence to PNE by best/better response is guaranteed (Monderer et al. 1996). Although the IPG is not always a potential game, it still shares many similarities. We make an analysis of convergence of the IPG in Section 5.3.3. Furthermore, in Chapter 6, we describe experimental results that aim to evaluate convergence properties by better-response dynamics in concrete domains that do not meet the conditions from the above Theorem 5.2.3. We note that the IPG is designed to be applicable to a wide range of real problems and this is the reason why we considered all the elements in the agents cost functions, which makes our model more complete.

In the following section, we take advantage of the potential game properties of the IPG in order to compute a PNE with better-response dynamics. We also analyze what happens when the IPG is not a potential game and we show that there still are positive properties.

5.3 Better-Response Planning Strategy

In this section, we explain the ***Better-Response Planning Strategy (BRPS)*** approach which is applied to solve the IPG. In the following subsections, we present the BRPS process, its search procedure, we prove the convergence of BRPS to a PNE, and we make a discussion about the complexity of BRPS in the IPG.

5.3.1 BRPS Process

Better-response dynamics draw upon the properties defined for best-response dynamics. Particularly, we know that any finite potential game (Monderer et al. 1996) will converge with *best-response dynamics* to a PNE regardless of the cost functions (e.g., they do not need to be monotonic). Moreover, it is not even necessary that agents best respond at every step since best-response dynamics will still converge to a PNE in a finite number of steps as long as agents deviate to a *better response* (Shoham et al. 2009, Chapter 6). Additionally, a better-response strategy can be implemented by an agent by randomly sampling another plan until one is found with a lower cost than the current plan's and hence, it does not require that the agent knows the cost of every plan in its search space (Friedman et al. 2001). In our planning context, we use better response instead of best response since agents do not need to find the best plan in each iteration which may be computationally intractable.

However, the task is already hard for an agent, as we pointed out in Theorem 5.2.2.

Our BRPS is a process in which each agent i iteratively revises its plan π_x^i in the joint plan Π to switch to another plan π_y^i which integrated in Π^{-i} reports a better utility than π_x^i for i. Before starting the process, an arbitrary order between the agents in $\mathcal{AG}$ is established. Moreover, an empty joint plan $\Pi = \varnothing$ is set. During the process, agents must better respond in each iteration. If an agent i is not able to come up with a better-cost plan, it does not change its plan. When no agent modifies its plan within a complete iteration because none of them can better respond, the better-response strategy has reached a convergence point in which the current joint plan is a PNE.

Table 5.1: Example of two agents with conflicts. PNE in bold

	π_1^2	π_2^2	π_3^2	π_4^2
π_1^1	$-2cc^1-1, -2cc^2-1$	$-cc^1-1, -cc^2-2$	**-1, -3**	$-cc^1-1, -cc^2-4$
π_2^1	$-cc^1-2, -cc^2-1$	**-2, -2**	-2, -3	$-cc^1-2$, $-cc^2-4$
π_3^1	**-3, -1**	-3, -2	-3, -3	$-cc^1-3$, $-cc^2-4$
π_4^1	$-cc^1-4$, $-cc^2-1$	$-cc^1-4$, $-cc^2-2$	$-cc^1-4$, $-cc^2-3$	**-4, -4**

Let us take a simple IPG example with two agents (1 and 2) and four plans per agent (π_1^1 to π_4^1; and π_1^2 to π_4^2). Table 5.1 represents an IPG example in its normal-form in which $costP(\pi_1^1) = costP(\pi_1^2) = 1$, $costP(\pi_2^1) = costP(\pi_2^2) = 2$, $costP(\pi_3^1) = costP(\pi_3^2) = 3$, and $costP(\pi_4^1) = costP(\pi_4^2) = 4$. The cells in Table 5.1 are the 16 joint plans which can be generated by the combination of the agents' plans along with the utility for each agent. The terms cc^1 and cc^2 denote the cost of the penalty/tax that agents 1 and 2 are charged, respectively, for the unsolvable conflicts in each joint plan. Table 5.1 shows 7 solution joint plans and the ones in bold are PNE. Assuming the BRPS process may arrive to the joint plan $\langle \pi_4^1, \pi_4^2 \rangle$ with utilities (-4,-4), BRPS will not yield a different joint

plan because no agent is able to come up with a better plan without conflicts given the plan of the other and thereby none of them can improve its utility. The joint plan $\langle \pi_4^1, \pi_4^2 \rangle$ is PNE but it is not a PO solution whereas the rest of PNE plans are all PO solutions. Consequently, better-response dynamics cannot guarantee PO solutions.

From the agents perspective, the BRPS process works as follows:

- An arbitrary order of agents in $\mathcal{AG}$ is established. BRPS incrementally builds an initial joint plan, $\Pi = \langle \varnothing, \ldots, \varnothing \rangle$, $\Pi = \langle \pi^1, \varnothing, \ldots, \varnothing \rangle$, $\Pi = \langle \pi^1, \pi^2, \varnothing, \ldots, \varnothing \rangle$ and so on following the established order. This construction follows a similar procedure as explained below except that agent i has no previous upper cost bound.
- In one iteration, an agent i performs the following steps:
  1. it analyzes the cost of its current plan π_x^i in the joint plan as specified in Equation 5.6 and sets $upper^i = costTotal(\pi_x^i, \Pi^{-i})$.
  2. it starts a planning search process to obtain a different plan, say π_y^i, that achieves $\mathcal{G}^i$. During search, a tree, where nodes represent an incrementally integration of the actions of π_y^i within Π^{-i}, is created. Every node is evaluated according to Equation 5.6 and if the cost is greater or equal than $upper^i$ then the node is pruned. Otherwise, when the node already holds all of the actions of the plan π_y^i and if $costTotal(\pi_y^i, \Pi^{-i}) < upper^i$, then the search stops because a better response has been found. In this case, $\Pi' = \langle \pi^1, \ldots, \pi_y^i, \ldots, \pi^n \rangle$ is returned.

3. in case the search space is exhausted and no better plan is found (we note plans are pruned by $upper^i$), agent i does not change its plan π_x^i in Π since i is in best response.

- When no agent in $\mathcal{AG}$ modifies its plan in a complete iteration, better-response dynamics has reached a convergence point and the current joint plan is a PNE.

5.3.2 Search Procedure

From a planning perspective, it is worth noting that an agent i does not pre-compute the complete set of plans that solve the task $\mathcal{T}^i$. An agent in BRPS implements an individual A* search procedure that allows the agent to dynamically generate progressively better responses and integrate them into the current joint plan.

In one iteration of BRPS, agent i calculates $upper^i = costTotal(\pi_x^i, \Pi^{-i})$ as the cost of its current proposal in the joint plan, removes π_x^i, and autonomously launches an A* search to find and integrate a better response π_y^i into the joint plan. The root node of the search tree contains a joint plan which is defined as the composition of Π^{-i} and an empty partial-order plan of agent i: $\pi_{y_0}^i = \langle \Delta^i = \emptyset, \prec \rangle$. We will denote such a combination as $\Pi^{-i} \circ \pi_{y_0}^i$.

At each level of the search tree, a node incorporates one action over its parent node and inter-agent conflicts are solved, if possible. Given the root node $\Pi^{-i} \circ \pi_{y_0}^i$, its successor nodes will contain $\Pi^{-i} \circ \pi_{y_1}^i$, where $\pi_{y_1}^i = \langle \Delta = \{\alpha_1^i\}, \prec \rangle$; a successor of $\Pi^{-i} \circ \pi_{y_1}^i$ will be $\Pi^{-i} \circ \pi_{y_2}^i$, where $\pi_{y_2}^i = \langle \Delta = \{\alpha_1^i, \alpha_2^i\}, \prec \rangle$; and so on until a node which contains $\Pi^{-i} \circ \pi_y^i$ is found. In other words, each node of the tree successively adds and consistently supports a new action until a node that contains a complete plan π_y^i that achieves $\mathcal{G}^i$ is found. Note that the

inter-agent orderings inserted in each node do not introduce any synergies between agents since, as explained in section 5.1, these elements are merely used for conflict and congestion resolution.

The search is aimed at finding a plan for agent i without conflicts. The procedure finishes once a conflict-free better response is found. If the agent finds a node that contains an element in conflict, the search keeps running until a conflict-free plan is found or the search space is exhausted. During search, the $upper^i$ cost bound is used to prune nodes that would not yield a solution better than the current one.

The heuristic search of BRPS draws upon some particular planning heuristics (Torreño, Sapena, et al. 2015) that enable agents to accelerate finding a conflict-free outcome. Assuming that the current plan of agent i in a joint plan is π^i and that the best-cost plan of agent i integrated in Π^{-i} has a total cost of $C^\star$, i might need as many iterations as $costTotal(\pi^i, \Pi^{-i}) - C^\star$ to reach the optimal solution, improving one unit cost at each iteration. However, the combination of heuristic search and the upper cost bound helps guide the search towards a better-response outcome very effectively.

5.3.3 Convergence to an Equilibrium

Better-response dynamics in an IPG may converge to a PNE joint plan which might possibly contain conflicts. In this section, we analyze the type of conflicts that may lead to this situation and we show that in the absence of those, BRPS converges to an IPG solution. We also analyze convergence in the non-potential version of the IPG.

Every potential game has at least one outcome that is a PNE, and better-response (or best-response) dynamics always converges to a PNE in any finite

potential game (Shoham et al. 2009, Chapter 6) (Nisan, Roughgarden, et al. 2007, Chapter 19).

Corollary 5.3.1. *Better-response dynamics in an IPG always converges to a PNE if the potential game property holds.*

As we explained in Theorem 5.2.3, the potential game property with the potential function Φ only holds under some assumptions. However, we must note that even without those assumptions and thus having the cost functions of the agents as defined in Equation 5.6 ($costTotal$, where the agents consider their own plans, congestions, unsolved conflicts, and delays by solved conflicts and/or congestions), the IPG with better-response dynamics will converge to a PNE in most cases.

Convergence to Conflict-free Joint Plans

In some problems, a joint plan with conflicts can be a PNE of the IPG and better-response dynamics could converge to this non-executable PNE joint plan. This is always caused by a multi-symmetric unsolvable situation among (at least) two agents which have a symmetric unsolvable conflict and none of them has a better response; that is, no alternative plan that improves u^i or u^j due to the existence of conflicts.

Definition 5.3.1. *There exists a* **Multi-Symmetric Unsolvable Situation** *(MSUS) between two agents i and j in an IPG if the following two conditions hold:*

1. *there exists a symmetric unsolvable conflict between a plan π^i and every plan of j that solves $\mathcal{T}^j$, and*

2. *there exists a symmetric unsolvable conflict between a plan π^j and every plan of i that solves $\mathcal{T}^i$*

In contrast to an unsolvable IPG (that would be the case when every plan of i contains a symmetric unsolvable conflict with every plan of j and vice versa), a MSUS states there is (at least) an IPG solution for the game but none of the agents is able to unilaterally find a better response if they get stuck in symmetric unsolvable conflicts. We note that, whereas a MSUS is defined between a pair of agents, it can affect any number of agents. However, the presence of a single MSUS between two agents is a sufficient condition to endanger the convergence to an IPG solution if agents get stuck in the specific plans involved in the MSUS.

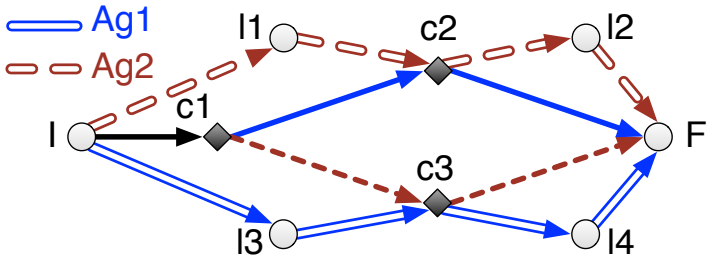


Figure 5.3: Multi-symmetric unsolvable situation example

Figure 5.3 shows a problem with a MSUS. Two agents, 1 and 2, are placed in location I and want to get to F. Agent 1 can only traverse solid edges and agent 2 dashed edges (except $I - c1$ which can be traversed by both agents). Locations $c1$, $c2$ and $c3$ can only be visited by one agent at a time, being permanently unavailable afterwards. Each edge has unitary cost. Agent 1 has two plans π_1^1 and π_2^1 with costs $costP(\pi_1^1) = 3$ and $costP(\pi_2^1) = 4$, corresponding to its inner and outer path, respectively. Similarly, agent 2 has two plans π_1^2 and π_2^2 corresponding to its inner and outer path, respectively, with costs $costP(\pi_1^2) = 3$ and $costP(\pi_2^2) = 4$. If both agents use their best plans, π_1^1 and

π_1^2, they will cause a symmetric unsolvable conflict at $c1$. If agent 2 switches to π_2^2, another symmetric unsolvable conflict will appear at $c2$. In the same way, if agent 1 switches to π_2^1, the symmetric unsolvable conflict will occur at $c3$. The only IPG solution is composed of π_2^1 and π_2^2, in which agents traverse the outer paths of Figure 5.3. This reveals that a better-response process can get trapped in a joint plan with conflicts which is PNE. This happens because a symmetric unsolvable conflict is only solvable through a bilateral cooperation, and in case of a MSUS like this, any alternative plan of one of the two agents also provokes a symmetric unsolvable conflict.

The strategies and utilities of this example are represented in Table 5.2, which is the normal-form of the IPG and includes all of the joint plans. A cell represents the utility of each agent in the joint plan formed by the plans of the corresponding row and column. The existence of a conflict in a joint plan entails a loss of utility of $-cc^i$ units. If one of the agents (or both) initiate the better-response process with their first plan, BRPS will converge to the non-executable joint plan with utilities $(-2cc^1 - 3, -2cc^2 - 3)$, which is a PNE. This happens because none of the agents is able to unilaterally improve its utility by switching to another plan. The utilities of the agents can only be improved if they bilaterally switch to π_2^1 and π_2^2, respectively. However, this can never happen in a sequential better-response dynamics.

Table 5.2: Multi-symmetric unsolvable situation. PNE in bold

	π_1^2	π_2^2
π_1^1	$\mathbf{-2cc^1-3, -2cc^2-3}$	$-2cc^1-3, -2cc^2-4$
π_2^1	$-2cc^1-4, -2cc^2-3$	$\mathbf{-4, -4}$

It should be noted that a MSUS is unlikely to occur in real-world problems as it features a very restricted scenario with several and fairly particular conflicts.

As shown in the example of Figure 5.3, the two agents block each other, not only for a plan but for all possible alternative plans since they only could reach a conflict-free joint plan through a bilateral plan switch. Hence, once these situations are identified, where BRPS could end up in a non-executable PNE, we can assure that in the absence of MSUS, if BRPS converges to a PNE it will be an IPG solution.

Corollary 5.3.2. *Better-response dynamics in an IPG without any* multi-symmetric unsolvable situation *always converges to a PNE if the potential game property holds, which is an IPG solution (conflict-free joint plan).*

As shown in Corollary 5.3.1, the IPG is a potential game (under some assumptions) with an associated ordinal potential function Φ of Equation 5.7 that guarantees convergence to a PNE with better-response dynamics. Thus, in the absence of MSUSs, agents will never get blocked in a symmetric conflict since, if an agent cannot solve it, the other involved agent will address the conflict. Therefore, agents will progressively reduce their costs by solving conflicts and improving their utility until converging to a PNE which is an IPG solution (conflict-free joint plan). In other words, if a game does not present MSUSs, only conflict-free joint plans can be PNE. Additionally, in the absence of MSUS, if BRPS converges to a PNE in the non-potential version of the IPG, then the PNE will be an IPG solution.

Convergence in the Non-Potential IPG Version

Better-response (or best-response) dynamics in the IPG may cycle only by the combination of the individual agents plans cost and congestion cost. For instance, if an agent i improves its cost by switching its plan to one that provokes a congestion to other agents, and the cost decrease of i does not compensate the cost increase of the other agents in congestion (reflected by Φ), the poten-

tial game property is broken. When the IPG is not a potential game, situations like the example we described may provoke cycles and better-response dynamics would never converge. However, it is not really common to find domains in which such cycles appear easily, as we will show in the experiments of Chapter 6.

To analyze what happens in the non-potential IPG version, in which all the cost elements of $costTotal$ are considered, we turn to the concept of a *sink equilibrium* (Goemans et al. 2005). We define a state graph $G = (V, E)$, where V are the states of the game (strategy profiles or joint plans Π in the IPG), and E are better or best responses, that is, an agent i has an arc from one state Π to another state Π' if it has a better/best response from Π to Π'. The evolution of game-play is modeled by a random path in the state graph, similarly to extensive-form games with complete information. Such a random path may converge or may not converge to a PNE, but it surely converges to a *sink equilibrium* (which may be or may not be a PNE). If we contract the strongly connected components of the state graph G to singletons, then we obtain an acyclic graph. The nodes with out-degree equal to zero are named sink nodes, that is, nodes with no out-going arcs in G. These nodes correspond to states of sink equilibria since random best/better-response dynamics will eventually converge to one of those (and will never leave it) with probability arbitrarily close to 1 (Goemans et al. 2005). Therefore, we announce the following proposition:

Proposition 5.3.1. *Random better(best)-response dynamics in an IPG without any* multi-symmetric unsolvable situation *will eventually converge to a sink equilibrium, which is a conflict-free joint plan.*

Proof. Similarly to Corollary 5.3.2, in the absence of MSUSs, agents will progressively reduce their costs by solving conflicts and improving their utility

until converging to a sink equilibrium because they would never get blocked in a symmetric conflict. A sink equilibrium is always a conflict-free joint plan since, in an IPG without MSUSs, all the conflicts of a joint plan can be avoided. Only conflict-free joint plans can be sink equilibria, so convergence to them is guaranteed. However, a sink equilibrium is not necessarily an IPG solution so it is not necessarily either a NE solution. □

Despite a sink equilibrium is not as *strong* as a PNE, we remark that, in most cases, random better-response dynamics may converge to a sink equilibrium which may be also a PNE. This is an important result in the IPG because even without the potential property which guarantees convergence, we can almost assure convergence. Furthermore, in the absence of MSUSs, the *equilibrium* achieved will always be a conflict-free joint plan. All these promising results will be reflected in the experiments of Chapter 6.

5.3.4 Complexity of Better Response in an IPG

In this subsection, we discuss the complexity of using better-response dynamics in an IPG, considering both the planning complexity and the complexity of computing a NE in a potential game.

The class of Polynomial Local Search problems (PLS) is an abstract class of all local optimization problems which was defined by Johnson et al. 1988. Examples of PLS-complete problems include traveling salesman problem, or maximum cut and satisfiability. Finding a NE in a potential game is also PLS-complete if the best response of each player can be computed in polynomial time (Fabrikant et al. 2004). Moreover, the lower bound on the speed of convergence to NE is exponential in the number of players (Hart et al. 2010). This is a lower complexity than finding a NE in a general-sum game as the IPG, which is PPAD-hard as we showed in Theorem 5.2.1.

While these are good news for the IPG in general, we note that computing a strategy for an agent implies to plan, which is PSPACE-complete in the general case (Bylander 1994), as we pointed out in Theorem 5.2.2. However, planning complexity can be lower for some planning domains as it is shown by Helmert 2003. Specifically, while bounded (length) plan existence is always NP-complete, non-optimal plans can be obtained in polynomial time for a transport domain without fuel restrictions (i.e., LOGISTICS, GRID, MICONIC-10-STRIPS, and MICONIC-10-SIMPLE). In contrast, optimal planning is always NP-complete. This is one of the reasons why the BRPS approach uses better-response dynamics instead of best-response dynamics because in terms of planning complexity it is easier to compute a non-optimal plan with satisficing planning.

Nevertheless, the inclusion of the IPG in the PLS class is not possible unless we are able to guarantee a best response in polynomial time. In the BRPS approach, only a better response (non-optimal plan) can be computed in polynomial time. Then, we need to guarantee that a sequence of better responses leads the game to a NE. In this sense, a bounded jump improvement (Chien et al. 2011) must be guaranteed in order to ensure PLS-completeness of the IPG with the BRPS approach.

Proposition 5.3.2. *Computing a PNE of an IPG, in its potential game version, using better-response dynamics is PLS-complete if non-optimal plans can be computed in polynomial time and a better response minimum improvement is guaranteed.*

Proof. Let us take a standard transport domain without fuel restrictions like LOGISTICS, GRID, MICONIC-10STRIPS, or MICONIC-10-SIMPLE, for which a non-optimal plan can be computed in polynomial time, as specified in (Helmert 2003). If we use a satisficing planner which computes non-optimal solutions,

and the planning agents always have a minimum jump improvement in their better responses, then achieving a PNE which is an IPG solution is in PLS. □

This is a positive result since it guarantees that for some specific planning domains, the complexity of solving this planning and game-theoretic problem is PLS-complete, which is much better than common PSPACE-completeness of planning and PPAD-completeness of computing a NE for any general-sum game.

5.4 Conclusions

In this chapter, we have presented an approach to address non-cooperative MAP problems that feature planning conflicts and congestion issues. These are problems where agents wish to make their interests prevail but coordinating their strategic behavior with the others yields better solutions. The inclusion of individual cost functions for the agents reflects strategic behavior and a more realistic representation of self-interested planning agents for real-world problems. Despite quite a few real-life problems follow this behavioral pattern, the study of this type of problems has been mostly neglected.

We define a general-sum game named Interaction Planning Game (IPG) in which self-interested planning agents consider interactions (conflicts and congestions) as part of their cost, as well as the cost of their own plans. If an agent provokes a conflict that impedes another agent to execute its plan, such a dis-coordinated situation may be an equilibrium solution in non-cooperative MAP. However, it is undesirable since some agent would not be able to execute its plan. Hence, we apply taxation schemes to agents that provoke planning conflicts. The tax applied to conflicts incentivize agents to avoid them, and

thus, we prove that any equilibrium is an IPG solution (conflict-free joint plan) in the absence of any multi-symmetric unsolvable situation.

Regarding the potential version of the IPG, we show that convergence to a PNE is always guaranteed with better/best-response dynamics. When congestion interactions and the individual cost of a plan are considered, the IPG is a non-potential game. However, better/best-response dynamics will converge to a PNE in most cases, or otherwise they can still converge to a sink equilibrium. We also prove that any equilibrium is an IPG solution (conflict-free joint plan) in the absence of multi-symmetric unsolvable situations.

We analyze the complexity of using better-response dynamics in an IPG and concluded that, non-optimal plans can be computed under some conditions. This is much less costly than computing the best response or optimal plan because agents do not need to explore all their strategies. Additionally, computing a NE is also a hard task, but using better-response dynamics may reduce the complexity of such task. For these reasons, we show promising results towards PLS-completeness under some assumptions.

All in all, we believe that BRPS may have a reasonable execution time in many practical cases while it is capable of tackling real-world problems more accurately than other approaches. Hence, the aim of the next chapter is to perform a comprehensive experimental evaluation of BRPS to assess its performance and applicability to non-cooperative MAP problems.

Chapter 6

BRPS Experimental Evaluation

This chapter presents a comprehensive empirical evaluation of the performance of the BRPS approach presented in Chapter 5. In order to properly assess the behavior of BRPS in different contexts, the experiments presented in this chapter are structured in three different scenarios:

- Congestion scenario: this setting evaluates BRPS by means of the network routing domain, introduced in (Jonsson et al. 2011), which features congestion issues but does not include planning conflicts.
- Conflict scenario: a collection of problems from the CoDMAP benchmarks (Komenda et al. 2016) where selected and adapted to a non-cooperative context in order to evaluate BRPS in a setting where planning conflicts among agents may arise. These domains do not consider congestion issues.
- Combined congestion/conflict scenario: we designed an additional MAP domain, called Electric Autonomous Vehicles, to test the performance

of BRPS in non-cooperative MAP problems that include both planning conflicts and congestion issues.

The experimentation compares BRPS against the state-of-the-art BRP[1] solver (Jonsson et al. 2011), one of the few fully-functional non-cooperative MAP approaches in the literature. BRP presents the following features:

1. BRP is specifically designed to compute equilibria in *congestion games* (Rosenthal 1973), where the simultaneous use of a resource by multiple agents increases its cost. The agents' cost functions in BRP do not consider planning conflicts.

2. BRP requires an initial conflict-free joint plan Π_{ini} that is calculated offline by means of a cooperative MAP solver[2]. The joint plan comprises one plan per agent in $\mathcal{AG}$, such that Π_{ini} achieves the agents' goals, $\mathcal{G}^i, \forall i \in \mathcal{AG}$.

3. BRP is an iterative plan improvement model wherein agents best respond to the plans of the other agents while maintaining the conflict-free structure of the joint plan. This means that agents plan at each iteration, thus having a potentially unlimited amount of plans.

4. BRP applies best response instead of better response, which implies that agents use cost-optimal planning machinery for the individual plan generation that may be more costly (NP-complete) than non-optimal search (polynomial time in some cases) (Helmert 2003).

[1] We used the optimal version of Fast Downward (FD) (Helmert 2006) as the underlying individual planner of BRP, since it was the best-performing setting in our tests.

[2] Due to implementation limitations of BRP, we used the satisficing LAMA planner (Richter et al. 2010) to compute the initial conflict-free joint plan, instead of a cooperative MAP solver.

5. BRP is proved to be useful for improving an initial congested conflict-free joint plan, thus increasing the utility of the agents in scenarios that feature congestion interactions.

The aforementioned features reveal that BRP is directly comparable to BRPS. For this reason, the experimental tests presented in this chapter compare the behavior of our BRPS approach against BRP.

This chapter is structured as follows: firstly, we discuss some details concerning the implementation of BRPS. Section 6.2 presents the results of BRPS in the network routing domain, which focuses on congestion issues exclusively. In Section 6.3, BRPS is evaluated by means of several conflict-based CoDMAP domains adapted to a non-cooperative context. Finally, Section 6.4 presents our Electric Autonomous Vehicles domain and the results obtained by BRPS in this setting, which presents both congestion issues and planning conflicts. We analyze aspects such as the strategic behavior of agents in BRPS or the influence of the ordering of the agents in the solutions synthesized by BRPS. The last section presents a general discussion on the results.

6.1 BRPS Implementation Details

BRPS is implemented on top of a modified version of the MH-FMAP satisficing MAP solver (Torreño, Sapena, et al. 2015), which is used by BRPS agents to individually compute plans. BRPS draws upon the features of MH-FMAP, including its multi-agent data structures, communication infrastructure and message-passing protocols, privacy model (Torreño et al. 2014b), and heuristic functions (Torreño, Sapena, et al. 2015).

An agent i of BRPS uses MH-FMAP to individually synthesize plans (responses) that are integrated in the current joint plan Π^{-i}. The search space of an

agent i is pruned by using the cost of its previous response, π_x^i, as a bound. Additionally, the search process is efficiently guided by the heuristic functions of MH-FMAP (Torreño, Sapena, et al. 2015), which have been adapted to deal with the cost functions of the agents. Moreover, the individual planner of each agent can return plans with unsolved conflicts.

Finally, it is worth noting that we designed an extension to the MAP language presented in (Torreño et al. 2014b), which is based on the Planning Domain Definition Language (PDDL), to define the elements that can be implied in a congestion. The planner was modified to include specific reasoning mechanisms that interpret and use the congestion information of the PDDL input files.

6.2 Congestion Scenario: Network Routing Domain

This section compares the performance of our BRPS approach and BRP in a network routing domain proposed in (Jonsson et al. 2011), which features congestion issues among agents.

6.2.1 Experimental Setup

The network routing domain (Jonsson et al. 2011) roughly emulates congestion in computer networks. This domain consists of a set of `nodes`, connected to each other through a variable number of `links` [1-3]. An agent in this domain represents a data package that must be transported from an origin to a destination `node`. The origin and destination of an agent are always different nodes, but these are not exclusive for the agent.

In this domain there are no planning conflicts between the actions of the agents, but two or more agents can be involved in a congestion if they tra-

verse the same link simultaneously. The cost of traversing a link is unitary, as well as the cost of a one-time-step delay. An agent cannot introduce parallel actions.

The theoretical cost of traversing a link in a computer network is defined as $c_l(n) = L \cdot n + E^{\max(0, n - C(l))}$, where n is the number of agents using the link simultaneously, $L = 1$ is the linear component, $E = 2$ is the exponential component, and $C(l)$ is the capacity of the link. In this MAP domain, the cost of a congested link l increases linearly with the number of agents that traverse it concurrently, as long as this number is under the link capacity, $C(l)$, which ranges between 1 and 10 agents. If the number of agents in a link l exceeds $C(l)$, the congestion cost increases exponentially.

The PDDL encoding of the network routing domain includes a single action `move` (see Listing 6.1), which represents a package `agent` moving from a `node n1` to another `node n2` through `link l`. The cost for an agent that executes a `move` action depends on the cost of traversing the `link l`, which is defined via the `link-cost1` function (this function reports a unitary cost).

Listing 6.1: PDDL encoding of the `move` action

```
(:action move
 :parameters (?a - agent ?n1 ?n2 - node ?l - link)
 :precondition (and (= (at ?a) ?n1) (has-link ?n1 ?n2 ?l) )
 :effect (and (assign (at ?a) ?n2)
              (increase (total-cost) (link-cost1 ?l))))
```

However, if the `link l` is congested, the cost of the `move` action is redefined according to the cost of the link, $c_l(n)$, which depends on the number of agents simultaneously traversing it. As shown in Listing 6.2, $c_l(n)$ is modeled through

the PDDL congestion link-use: for 2 agents, the `link-cost2` function is applied, for 3 agents `link-cost3`, and so on.

Listing 6.2: Excerpt of the PDDL congestion `linkuse`

```
(:congestion linkuse
 :parameters (?l - link)
 :variables (?a - agent ?n1 ?n2 - node)
 :usage (move ?a ?n1 ?n2 ?l)
 :penalty (and
  (when (= (usage) 2) (increase (total-cost) (link-cost2 ?l)))
  (when (= (usage) 3) (increase (total-cost) (link-cost3 ?l)))
  ...
))
```

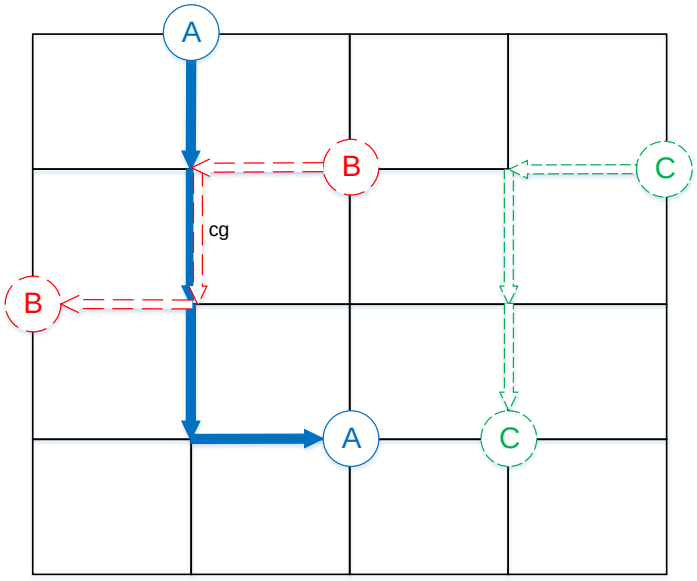


Figure 6.1: Network routing domain example

Figure 6.1 shows an example of application where three agents, named *A*, *B*, and *C*, traverse the network from an origin to a destination node. In this example, a congestion occurs in the link labeled as cg because agents *A* and *B* traverse it at the same time step $t = 1$. In the case of agent C, its path has no interaction with other agents.

6.2.2 Results

Tables 6.1 and 6.2 summarize the comparative results of BRP (Jonsson et al. 2011) and our BRPS approach in the network routing domain. The results show the different features of the solution joint plans obtained by each approach, such as the number of actions (acts), makespan or finish time of the joint plan (ms), and cost. We also show the number of iterations (iters) spent by BRP and BRPS to find the equilibrium solution plans, and the total planning time (time).

Table 6.1: Experimental results of BRP and BRPS for networks of 10 nodes and 5 to 40 agents

	BRP					BRPS				
Agents	Acts	Ms	Cost	Iters	Time	Acts	Ms	Cost	Iters	Time
5	**9.0±2.49**	3.0±0.67	10.8±3.58	2.5±0.53	0.637±0.17	9.1±2.56	**2.7±0.67**	**10.3±3.56**	2.5±0.53	**0.063±0.02**
10	18.9±2.33	3.8±0.42	**23.6±3.95**	**3.0±0.47**	1.686±0.28	**18.8±2.30**	**3.1±0.74**	24.3±3.92	3.1±0.53	**0.130±0.02**
15	27.4±4.38	3.7±0.67	**37.4±8.11**	**3.6±0.52**	3.481±0.75	**27.0±3.97**	**3.1±0.74**	39.7±9.94	3.7±0.48	**0.256±0.11**
20	38.0±4.14	4.3±0.67	59.2±12.22	**3.8±0.63**	5.318±1.05	**36.7±2.71**	**3.9±0.48**	**59.1±12.90**	3.9±0.57	**0.435±0.36**
30	61.2±6.86	5.0±0.67	**123.6±25.62**	**4.2±1.03**	10.715±3.35	**59.7±5.19**	**4.9±0.57**	129.4±34.81	4.4±0.47	**1.300±0.65**
40	**87.0±15.69**	**5.6±1.71**	**249.5±77.34**	**4.7±1.49**	19.387±8.82	88.7±12.21	5.8±1.63	251.2±79.87	5.1±1.42	**2.161±0.97**

The first experiment, shown in Table 6.1, tests the performance of BRP and BRPS in networks of fixed size and a variable number of agents. More precisely, a row in Table 6.1 shows the average results of 10 randomly generated networks of 10 nodes for 5 to 40 agents. The capacity of a link ranges between 1 and 10 packages (agents), and the origin and destination nodes of the agents are also randomly defined.

Regarding performance, it is worth noting that the number of iterations scales up linearly with the number of agents (see Figure 6.2). The reason behind this result is that a higher number of data packages (agents) in a computer network raises the number of congestion issues. This increases the complexity of the problem, since avoiding a large number of congestions to reach an equilibrium generally requires a high amount of iterations, which also causes an impact on the overall planning time. In most problems, both approaches require a similar number of iterations to yield solutions. However, since BRPS uses better-response dynamics, it requires more iterations to converge in some instances, because unlike BRP, agents do not necessarily propose the best possible response in an iteration.

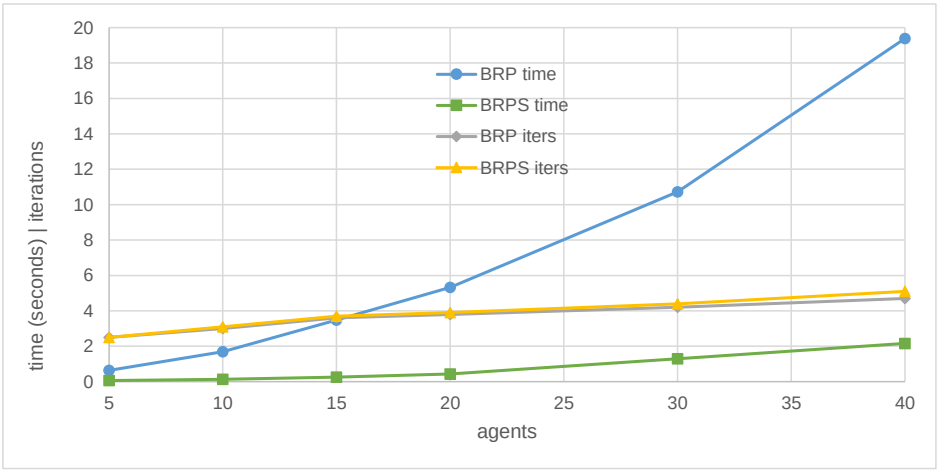


Figure 6.2: Planning time (in seconds) and iterations of BRP and BRPS in the experiments of Table 6.1 for 5 to 40 agents

In terms of planning time (see Figure 6.2) BRPS performs significantly better than BRP, since the latter approach uses a centralized planner, which is not optimized for MAP tasks, to synthesize responses. Additionally, BRP uses opti-

mal search, which is in general computationally harder than our sub-optimal approach.

In terms of joint plan quality (number of actions, makespan and cost), both approaches present very similar results, as shown in Table 6.1 and Figure 6.3. It is important to note that, as the number of agents increases, the quality of the solution joint plans decreases, since congestion issues in a network are more frequent if the number of agents (data packages) is high. Agents can either incur in congestions, which penalizes their costs, or take alternative longer paths to avoid them, thus jeopardizing the number of actions and makespan of the solution joint plans.

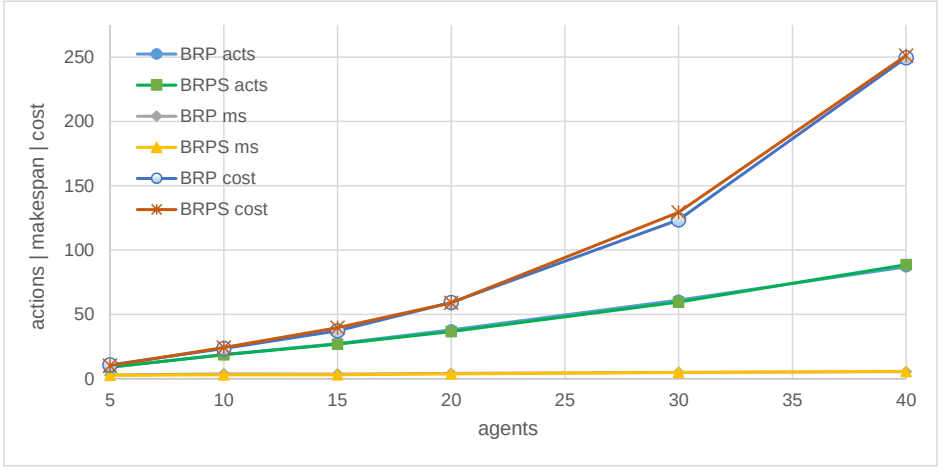


Figure 6.3: Average number of actions, makespan, and cost of BRP and BRPS in the experiments of Table 6.1 for 5 to 40 agents

While BRP starts the problem-solving process with an initial joint plan, BRPS executes the better-response dynamics from scratch. For this reason, BRPS and BRP yield slightly different solution joint plans with different global costs. Best-response and better-response dynamics can be interpreted as a search of local maximums, which means that there may be different equilibrium so-

lutions that report notably different costs to a given agent. However, irrespective of the use of better or best-response dynamics, the solution joint plan depends not only on the mechanism applied to build the initial joint plan, but also on the ordering of the agents in the game.

For example, let us assume a problem with two agents, named 1 and 2, and a network of unitary cost links where a congestion implies a 4-unit cost. Both agents have an optimal path of three links (3 cost units). Agent 1 has a secondary path of four links (4 cost units), and agent 2 has an alternative path of five links (5 cost units). If agent 1 puts forward its plan first, it will choose its optimal 3-cost-unit plan. Then, agent 2 tries to put forward its 3-cost-unit plan, which would cause a congestion, since the route for agent 2 shares a link with agent 1. This congestion would raise the cost of the plan from 3 to 6 units (two 1-cost-unit links and one congested 4-cost-unit link). For this reason, agent 2 selects instead its 5-cost-unit plan, which does not cause any congestion. This results in a solution joint plan with associated costs (3, 5). However, if agent 2 puts forward its plan before agent 1, the game would yield a solution joint plan with costs (4, 3). Both solutions are equilibria that will be found through better and best-response dynamics. However, the latter one has lower global cost. This proves that the ordering of the agents in the game is a key aspect that influences the final solution, which explains the variability of the plan quality results in these experiments.

The second experiment aims to study the influence of the network size (number of nodes) in the results. Table 6.2 shows the average results for 10 randomly generated networks of 5 to 100 nodes and 10 agents. In terms of performance, one can observe that the number of iterations of BRP and BRPS remains relatively stable, while planning time increases with the network size (see Figure 6.4). Regarding planning time, BRPS clearly outperforms BRP, which scales up poorly, as shown in Figure 6.4. Since both approaches spend

Table 6.2: Experimental results of BRP and BRPS for networks of 5 to 100 nodes and 10 agents

	BRP					BRPS				
Nodes	Acts	Ms	Cost	Iters	Time	Acts	Ms	Cost	Iters	Time
5	**16.4±4.65**	3.5±0.97	35.7±12.01	2.9±0.57	1.222±0.26	16.7±3.40	3.4±0.84	35.2±15.32	2.9±0.53	**0.086±0.02**
10	18.9±2.33	3.8±0.42	23.6±3.95	3.0±0.47	1.714±0.28	18.8±2.30	**3.1±0.74**	24.3±3.92	3.2±0.53	**0.138±0.02**
15	**24.1±4.56**	4.6±0.52	27.1±7.42	3.0±0.00	2.606±0.35	24.4±4.40	**4.4±0.70**	27.2±7.35	3.1±0.42	**0.187±0.02**
20	22.8±2.66	4.3±048	25.4±3.86	3.1±0.57	3.343±0.91	22.8±2.49	**3.8±0.63**	25.4±3.92	3.0±0.53	**0.207±0.03**
30	**26.6±3.20**	4.6±0.52	27.8±4.52	2.6±0.70	4.981±1.91	26.8±3.22	4.4±0.97	28.2±5.07	2.7±0.32	**0.250±0.03**
40	30.8±3.22	5.1±0.57	32.8±2.94	3.0±0.47	10.326±2.32	**30.4±3.34**	4.8±0.42	32.8±3.33	3.2±0.52	**0.286±0.04**
50	28.6±2.91	4.8±0.42	29.6±3.44	2.6±0.52	12.165±4.03	28.5±3.03	4.6±0.52	29.7±3.65	2.5±0.32	**0.362±0.14**
60	**31.5±2.32**	5.0±0.67	32.3±2.26	2.3±0.48	12.931±4.24	31.7±2.45	4.9±0.53	32.3±1.83	2.2±0.32	**0.473±0.16**
70	32.0±3.71	5.1±0.57	32.6±3.27	3.0±0.47	25.759±5.31	32.0±3.71	4.9±0.53	32.8±3.22	3.3±0.53	**0.420±0.17**
80	31.6±1.96	5.0±0.00	31.8±1.75	2.6±0.52	25.608±8.78	31.4±2.07	4.5±0.53	32.0±1.70	3.0±0.32	**0.443±0.12**
90	34.3±4.47	5.7±0.67	35.1±4.82	2.6±0.52	36.854±14.18	34.1±4.28	5.4±0.84	35.1±4.95	2.8±0.42	**0.518±0.20**
100	34.6±4.25	5.7±0.67	34.6±4.25	2.5±0.53	37.221±13.00	34.6±4.25	5.2±0.63	34.6±4.25	2.6±0.48	**0.469±0.07**

roughly the same number of iterations across all the problems, we can conclude that BRPS requires a significantly lower planning time per iteration than BRP. The performance differences become more apparent in large networks that exceed 30 nodes.

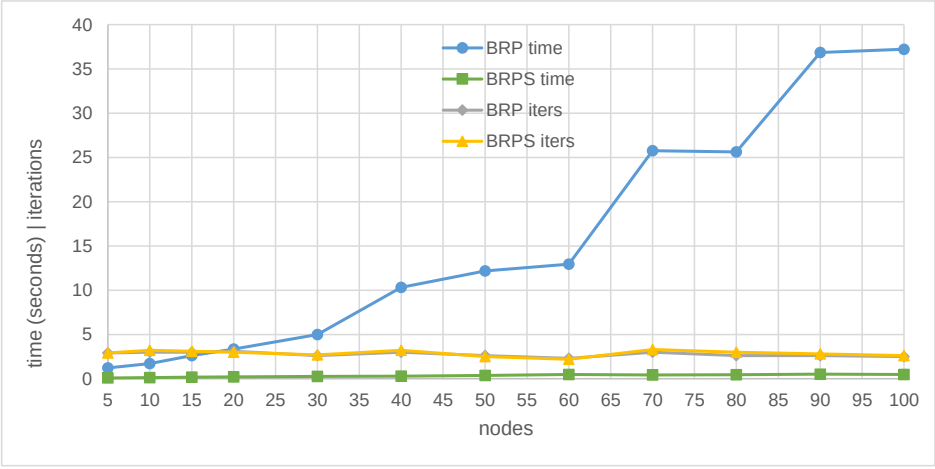


Figure 6.4: Average planning time (in seconds) and iterations of BRP and BRPS in the experiments of Table 6.2 for networks from 5 to 100 nodes

The quality of the solution joint plans remains stable regardless of the size of the network (see Figure 6.5). Note that the number of actions and makespan of the solutions increase slightly with the number of nodes, because the data packages (agents) traverse longer paths to reach their destinations. In 5-node networks, the average cost of the solution joint plans is notably higher than the subsequent cost results in Table 6.2. This is explained by the reduced size of these networks, which causes many unavoidable congestions, thus penalizing the plan cost.

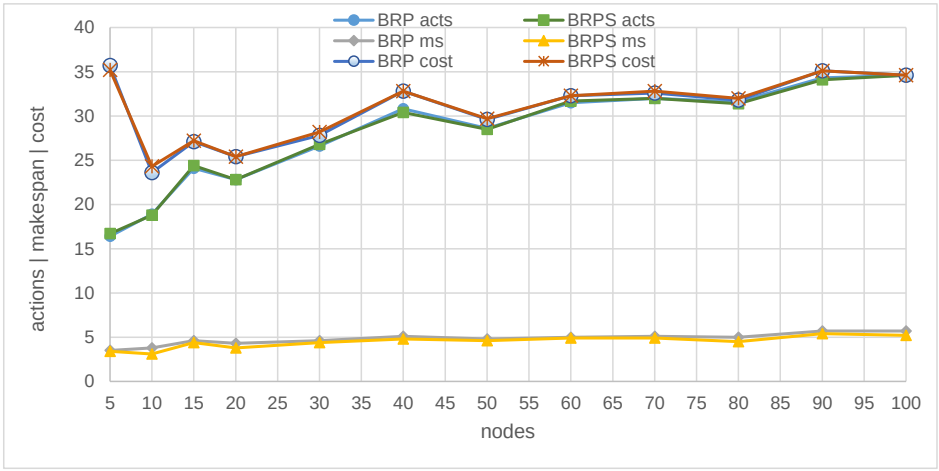


Figure 6.5: Average number of actions, makespan, and cost of BRP and BRPS in the experiments of Table 6.2 for networks from 5 to 100 nodes

In summary, agents in this congestion-focused setting are significantly affected by congestions in networks where these issues are not avoidable. Incurring in a congestion penalizes the cost of the solution joint plan, while deviating from the optimal route to avoid it affects the number of actions and makespan of the solution. The results reveal that BRPS and BRP achieve similar results in terms of cost and plan quality, since the strategies they use to reach an equilibrium perform similarly in this setting, which does not consider

planning conflicts. However, our BRPS approach significantly outperforms BRP in terms of planning time, proving to scale up much better, particularly in problems that feature complex computer networks or large amounts of agents.

6.3 Conflict Scenario: CoDMAP Domains

The second test compares BRPS and BRP over a set of non-cooperative MAP problems based on several well-known domains of the 2015 Competition of Distributed and Multi-Agent Planners (CoDMAP)[3] (Komenda et al. 2016).

The purpose of this test is to assess the performance of BRPS in a setting where planning conflicts may emerge among the participating agents. In other words, this test measures the ability of BRPS agents to circumvent conflicts and yield executable solution plans.

6.3.1 Experimental Setup

This benchmark includes a set of non-cooperative problems from four different CoDMAP domains that have been customized to maximize the appearance of conflicts among agents:

- *Driverlog (dl)*: in this transportation domain, driver agents compete for the use of trucks that allow them to deliver packages. In our non-cooperative version, a location can only be occupied by a single truck. Therefore, conflicts may arise if two agents try to drive concurrently to a given location.
- *Floortile (fl)*: in this domain, a set of robots paint a grid of floor tiles in different colors. A robot can only paint the tile that is above or below its

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

location, but it can move both vertically and horizontally. Once a tile is painted, it cannot be traversed, and only one robot can be in a tile at a time.

- *Rovers (rv)*: this domain models a set of rover agents that must collect a variety of samples (rocks, soil samples, and pictures). In order to maximize the number of conflicts among agents, a waypoint can only be traversed by one rover at a time, and agents must take turns to communicate the collected samples to the lander.

- *Zenotravel (zt)*: in this domain, agents are aircrafts that must transport persons over a set of cities. In order to maximize the appearance of conflicts, only one aircraft can be on a given city at a time. Additionally, a person can only be transported by an a priori defined aircraft.

6.3.2 Results

Table 6.3[4] summarizes the results of BRP and BRPS regarding solution joint plan quality (number of actions, makespan, and cost) and performance (number of iterations and computation time). For each MAP domain in this test, we defined 5 non-cooperative problems of growing complexity. The rightmost number of a problem name represents the number of agents that take part in the problem (for instance, *dl1-2* refers to the first *driverlog* problem, which features 2 agents).

As in the congestion scenario (see Section 6.2), BRPS clearly outperforms BRP in terms of computation time, despite the fact that our approach requires an additional iteration to converge in some instances. In general, BRPS is two

[4] All the tests were conducted on a single machine with an Intel Core i7-3770 CPU at 3.40GHz and 8 GB RAM. Each test was run within a time limit of 1800 seconds.

Table 6.3: Results of BRP and BRPS in CoDMAP problems

	BRP					BRPS				
Prob-Ag	Acts	Ms	Cost	Iters	Time	Acts	Ms	Cost	Iters	Time
dl1-2	9	6	11	2	1.49	9	6	11	2	**0.31**
dl2-2	11	7	13	2	1.31	11	7	13	2	**0.32**
dl3-3	16	7	18	2	2.78	16	7	18	2	**0.58**
dl4-3	19	8	21	2	2.99	19	8	21	2	**0.63**
dl5-3	18	7	19	2	3.33	18	7	19	2	**0.65**
ft1-2	8	4	8	2	1.80	8	4	8	2	**0.47**
ft2-3	11	4	11	2	2.16	11	4	11	2	**0.50**
ft3-4	15	6	15	2	2.38	**13**	**4**	**14**	2	**1.23**
ft4-5	19	10	20	**2**	4.85	19	10	20	3	**2.17**
ft5-5	22	10	23	2	5.20	22	10	23	2	**1.55**
rv1-3	10	5	12	**2**	1.52	10	**4**	**11**	3	**0.85**
rv2-4	15	6	18	**2**	2.15	**14**	**4**	**15**	3	**0.98**
rv3-5	15	6	20	3	4.23	15	**4**	**16**	3	**1.57**
rv4-6	17	7	27	3	5.44	17	**4**	**18**	3	**2.05**
rv5-6	**23**	9	31	3	6.26	24	**7**	**24**	**2**	**1.66**
zt1-2	**12**	7	17	3	4.79	13	**6**	**13**	3	**0.78**
zt2-2	20	12	23	2	10.35	**17**	12	**20**	2	**0.79**
zt3-3	**22**	**12**	**42**	**2**	24.32	28	19	45	4	**7.45**
zt4-3	**28**	**11**	**33**	**2**	24.50	38	21	52	4	**15.52**
zt5-4	35	**16**	**43**	3	621.27	**28**	22	59	**2**	**14.34**

to four times faster than BRP, being even two orders of magnitude faster in problem *zt2-2*.

As previously discussed, BRP starts the best-response dynamics from an initial joint plan built by a centralized planner, while BRPS computes solution joint plans from scratch. The execution time results in Table 6.3 do not consider

the time spent by BRP to synthesize this initial joint plan, so the performance difference between both approaches is even higher.

Regarding joint plan cost, BRPS tends to obtain better solutions, since, in contrast to BRP, the utility functions of BRPS agents explicitly consider planning conflicts. This makes BRPS more effective at minimizing the makespan of the solution plans in this conflict-based scenario, which also benefits the overall cost of the solutions.

In *driverlog*, both approaches exhibit a notable performance, obtaining solution joint plans of the same quality in the five tested problems. While both planners manage to converge in 2 iterations in all the *driverlog* problems, BRPS is one order of magnitude faster than BRP, which once again proves the practical efficiency of its better-response dynamics.

In *floortile*, BRPS and BRP obtain similar results, with the notable exception of problem *ft3-4*, where BRPS yields a superior solution joint plan in terms of makespan and cost. Despite the complexity and high amount of interactions among robot agents in the *floortile* problems (note that the robots move through relatively tight maps where the tiles become unusable once painted), BRPS agents manage to optimize the quality of the solution plan while avoiding the conflicts that arise during the better-response dynamics. BRP agents, on the other hand, are unable to improve on the quality of the initial joint plan in most instances.

BRPS significantly outperforms BRP in the *rovers* domain, attaining better-quality solutions in all five problems. Planning conflicts in this domain are easily solvable, since they are basically related to the occupancy of the different waypoints. Once a planning conflict is detected, a rover agent can either take an alternative route or wait until the affected waypoint becomes traversable again. In this context, the results show that BRPS agents behave

strategically to minimize the length of their plans, thus effectively optimizing the makespan and cost of the solutions. Regarding BRP, we must note that its underlying centralized planner does not properly parallelize the actions of the agents, and hence, it yields longer solution plans than BRPS, which also jeopardizes their overall cost. Moreover, the solution plans of BRP are in part determined by the initial joint plan obtained prior to the execution of the best-response dynamics. In most cases, BRP agents are unable to deviate from this initial joint plan without causing new conflicts. Finally, it is worth noting that BRPS solutions properly balance the agents' costs; that is, our approach attains PNEs in which none of the agents is particularly penalized.

The *zenotravel* domain presents mixed results: BRPS obtains better solution joint plans in two of the problems, while BRP outperforms BRPS in the three remaining instances. Since an aircraft agent can directly travel between any pair of cities, *zenotravel* presents a wider array of alternatives to solve conflicts that the rest of domains in the benchmark. In this context, the planning strategy of BRP is particularly efficient, since agents manage to clearly improve on the initial joint plan during the best-response dynamics. This explains the relatively better solutions obtained by BRP in some of the problems. Despite yielding worse solutions in some *zenotravel* instances, BRPS is noticeably faster than BRP; in particular, BRPS solves *zt5-4* in 14 seconds, while BRP spends more than 10 minutes to synthesize a solution.

In conclusion, BRPS proves to be the superior approach in this conflict-based scenario, often yielding better solution plans in terms of actions, makespan and cost, while requiring much lower execution times than BRP. The performance of BRP in this context is limited by some aspects of its strategy: on the one hand, BRP relies on an initial joint plan synthesized by a centralized planner. Generally, this initial plan is not properly balanced among the agents, which may cause a utility loss. On the other hand, BRP agents do not reason

about conflicts, which makes difficult for them to improve on the initial joint plan. In contrast, BRPS properly circumvents planning conflicts while minimizing makespan and cost. Moreover, the use of suboptimal planning machinery in BRPS clearly benefits its computation times.

6.4 Combined Scenario: Electric Autonomous Vehicles Domain

The final test evaluates BRPS on a complex scenario that combines both congestion issues and planning conflicts. For this purpose, we defined a non-cooperative MAP domain where several autonomous taxi companies (agents) seek their own benefit without behaving in a strictly competitive manner. This domain models a real-world non-cooperative MAP problem in the context of a clean, coordinated and harmonic smart city.

Section 6.4.1 defines and motivates our Electric Autonomous Vehicles (EAV) domain, while Section 6.4.2 presents the experimental results obtained by BRPS and BRP on the EAV domain.

6.4.1 Case Study: Electric Autonomous Taxis in a Smart City

Sustainable and clean energy is one of the crucial aspects that define a smart city. Nowadays, the automobile industry is facing a conversion towards all-electric vehicles, and e-mobility has become one of the main objectives of both traditional car manufacturers and technological companies. However, the development of all-electric vehicles still faces some challenges like the deployment of a network of charging stations, limitations in the installation's power management or the widespread of slow-charging stations (Nigro et al. 2015).

On the other hand, autonomous (driverless) vehicles is also an emerging technology of growing interest in the context of smart cities, which is regarded as the future of the automobile industry. Advances in perception and algorithmic improvements in high-level reasoning capabilities, such as planning, predict a growing adoption of self-driving vehicles not limited only to public transportation (Stone et al. 2016).

The introduction of Electric Autonomous Vehicles (EAV) opens up a new scenario in transportation planning and poses key challenges in electric vehicle supply equipments such as the level of power demand (Wishart 2013). Thereby, it is important to avoid congestions due to excessive or peak-power demand. Moreover, car-to-car coordination needs to be applied in order to minimize accidents and rationalize traffic flow in densely-populated smart cities.

Let us suppose that a set of electric driverless taxi companies that operate on a smart city, individually design cab ride plans to carry passengers to their destinations, and recharge the batteries of their taxis when necessary. Each taxi company, which acts as a rational self-interested agent, designs a plan that maximizes its own benefit. However, at execution time, unexpected congestion issues caused by traffic jams or high occupancy rates in charging stations may hamper the predicted benefit of company agents' plans. This configures a multi-agent scenario where company agents must coordinate their plans with each other in order to maximize their benefits (Grosz et al. 1998). Therefore, taxi company agents are self-interested but behave on a coordinated manner, since a selfish behavior could prevent them from achieving their objectives.

Coordinated planning by self-interested agents is a desirable approach to prevent conflicts and unnecessary delays in the aforementioned scenario. For example, if multiple EAV need to recharge their batteries in a station that has

a single connector, a company may act strategically by forwarding its taxi into the station before the rest of implied EAV, thus forcing them to wait. By a priori coordinating their plans, the company agents can plan alternative routes for their EAV, so that they recharge their batteries in different stations, thus obtaining a joint outcome that reports better utilities to all the companies.

Through the use of MAP and the design of game-theoretic outcomes, driverless taxi companies operate as autonomous and *rational agents* that interact with each other to dynamically adjust the routes of their vehicles according to the planned routes of the rest of companies.

Domain Definition

In order to properly motivate our EAV case study, Figure 6.6 shows the area covered by several taxi companies in a European city. The route of a `taxi` is determined by the streets it traverses (black edges). A street is defined by the two `junctions` (gray circular nodes) it connects. Across the city, there are several `chargers` (green squares) where the taxis recharge their batteries.

A taxi `company` agent must coordinate its fleet of `taxis` to provide transport services to `passengers` that are located in different `junctions` and want a ride to specific destinations. A `company` agent plans the routes of its `taxis` across the network of streets in order to deliver the `passengers` in a cost-optimal way. Since energy management is a critical aspect of electrical vehicles, the course of action of a taxi `company` must include the necessary stops to recharge the batteries of its `taxi` fleet in the available `chargers` across the smart city.

This EAV domain was encoded with an extended version of the MAP language introduced in (Torreño et al. 2014a) that incorporates explicit support of congestion constraints. Taxi agents manage the following object types:

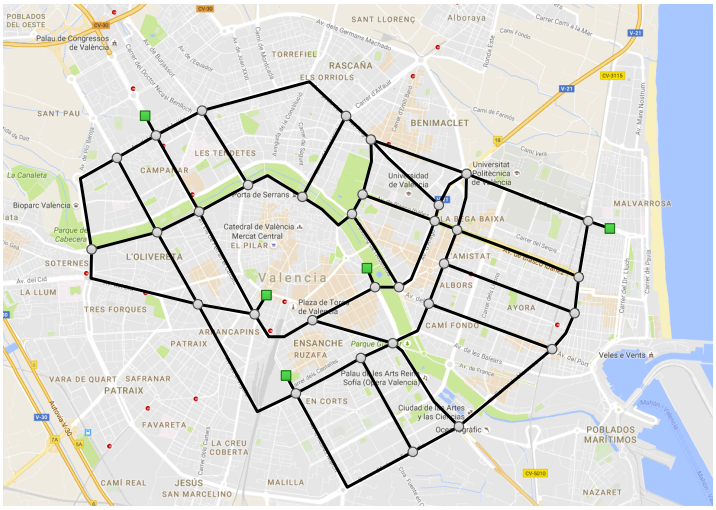


Figure 6.6: Smart city map example

- `company`: identifies the taxi `company` agents.
- `taxi`: identifies a `taxi` of a `company`.
- `location`: agents traverse two types of `locations`: street `junctions` and electric `chargers`.
- `passenger`: represents the clients that must be transported by the taxi companies.
- `level`: models the level of battery charge of an electric `taxi`.

The situations or states of the world are described through the following `:predicates` and `:functions`:

- `(traversing ?t - taxi) - location`: this function returns the current `location` of a `taxi`, either a street `junction` or a `charger`.

- (position ?p - passenger) - (either junction taxi): this function models the position of a passenger ?p, which is either waiting at a street junction or inside a taxi towards his/her destination.
- (street ?j1 ?j2 - location): this predicate indicates that there is a street which connects locations ?j1 and ?j2, so that a taxi which is traversing ?j1 can progress to ?j2.
- (free-taxi ?t - taxi): the taxi ?t is not occupied by a passenger if this predicate holds.
- (max-battery-capacity ?t - taxi ?l - level): the level ?l represents the maximum capacity of the taxi battery.
- (empty-charger ?c - charger): if true, the charger ?c is not occupied and can be accessed by a taxi.
- (current-charge-level ?t - taxi) - level: this function models the current level of charge of a taxi ?t.
- (destination ?p - passenger ?j - junction): this predicate represents the destination of a passenger ?p.
- (next-level ?l1 ?l2 - level): this predicate indicates that levels ?l1 and ?l2 are consecutive.

Taxi company agents individually plan the routes of their taxis by applying a set of planning *actions*:

- (drive ?t - taxi ?j1 ?j2 - junction ?l1 ?l2 - level): the taxi drives from junction ?j1 to junction ?j2 reducing its battery level from ?l1 to ?l2.

- `(charge ?t - taxi ?j - junction ?ch - charger ?n - network ?cl ?ml - level)`: the `taxi ?t` enters the `charger ?ch`, connected to `network ?n`, from `junction ?j` and charges its battery from its current `level`, `?cl`, to its maximum capacity, `?ml`.
- `(leave-charger ?t - taxi ?ch - charger ?j - junction)`: the `taxi ?t` leaves the `charger ?ch` and goes back to `junction ?j`.
- `(pick-up-passenger ?t - taxi ?p - passenger ?j - junction)`: the `passenger ?p` waiting at `junction ?j` gets into the empty `taxi ?t`.
- `(drop-passenger ?t - taxi ?p - passenger ?j - junction)`: the `passenger ?p` leaves the `taxi ?t` at his/her destination `?j`.

Since `taxis` act in the same environment, their activities may lead to a ***charging station occupancy conflict***. A `charger` is accessible by a single `taxi` at a time. When a `taxi` comes across an occupied `charger`, the `company` agent can either forward the `taxi` to a different `charger` (i.e. modify its plan), or make it wait until the occupying `taxi` leaves the `charger` (i.e. delay the charge action to circumvent the conflict).

The encoding of the `charge` operator includes a precondition `(empty ?ch)` and an effect `(not (empty ?ch))`, which effectively prevents other `taxis` from accessing the `charger` until the occupant leaves it through an action `leave-charger`, which has an effect `(empty ?ch)`. Therefore, a conflict will emerge if two `taxis` attempt to enter one particular `charger` at the same time or if there is already a taxi in the `charger`.

The traffic flow in a smart city may lead to congestions that directly affect the cost of the `taxis`' activities. In this case study, we identify two different types of congestions:

- ***Traffic jam congestion***. If several `taxis` drive simultaneously through a street between two `junctions`, traffic in such street will become less fluid, resulting in a traffic jam congestion. Consequently, the cost associated to the `drive` action of each `taxi` will increase. Therefore, agents should consider traffic congestions when selecting the routes of their `taxis`.

Listing 6.3: PDDL congestion `traffic-jam`

```
(:congestion traffic-jam
  :parameters (?j1 - junction ?j2 - junction)
  :variables (?t - taxi ?l1 - level ?l2 - level)
  :usage (drive ?t ?j1 ?j2 ?l1 ?l2)
  :penalty (and
           (when (= (usage) 2) (increase (total-cost)
                 (traffic-jam-cost-2 ?j1 ?j2)))
           (when (>= (usage) 3) (increase (total-cost)
                 (traffic-jam-cost-3 ?j1 ?j2)))))
```

The PDDL code of Listing 6.3 defines two different penalties: if two `taxis` traverse the street between `?j1` and `?j2` simultaneously, they will be penalized with a cost defined by the function `traffic-jam-cost-2`, while if three or more `taxis` are involved in the congestion, their owner `companies` receive a higher fee, represented by `traffic-jam-cost-3`.

- ***Electricity network congestion***. When many `taxis` intend to recharge their batteries simultaneously at chargers that are connected to the same electrical network, prices will raise due to a peak demand, thus leading to an electricity shortage. For this reason, `company` agents will be penalized if they get involved in an electricity network congestion.

Listing 6.4: PDDL congestion `electricity-network-overload`

```
(:congestion  electricity-network-overload
  :parameters (?n - network)
  :variables (?t - taxi ?j - junction ?ch - charger
                   ?cl - level ?ml - level)
  :usage (charge-battery ?t ?j ?ch ?n ?cl ?ml)
  :penalty (and
           (when (= (usage) 2) (increase (total-cost)
                 (network-cost-2 ?n)))
           (when (>= (usage) 3) (increase (total-cost)
                 (network-cost-3 ?n)))))
```

As shown in the PDDL code of Listing 6.4, we consider that an electricity network congestion appears when two or more `taxis` charge their batteries simultaneously in the same network. If only two `taxis` are connected to the electrical network, they are penalized with a cost defined by the function `network-cost-2`. In case that three or more `taxis` charge their batteries simultaneously in the same electrical network, the power demand will increase considerably, and `company` agents will receive a higher penalty, defined as `network-cost-3`.

In this scenario, where concurrent actions of self-interested agents can provoke congestions and conflicts, the best individual plan of an agent may not be the course of action that maximizes its utility in a joint plan. Moreover, a conflict makes the involved plans be non-executable. Therefore, agents are willing to give up their best individual plan for the sake of a safe joint plan that guarantees a stable execution of all the involved parties. For this reason, our BRPS model emerges as an appropriate mechanism to tackle route planning of electric autonomous vehicles.

Problem Example

This section analyzes a small example problem (see Figure 6.7) based on the EAV domain, in order to illustrate the behavior of BRPS. The example consists of three taxi companies, `Company1`, `Company2`, and `Company3`, each having a single vehicle (`t1 t2 t3`) and a passenger to transport (`p1 p2 p3`). There are four connected junctions, `j1` to `j4`, and two chargers, `c1` and `c2`, connected to the same electrical network, `n1`, and accessible from junctions `j1` and `j2`, respectively (see Figure 6.7). Taxis `t1` and `t3` start at `j1`, and `t2` starts at `j2`. The battery level of all taxis is `l0`, and their capacity is `l2`.

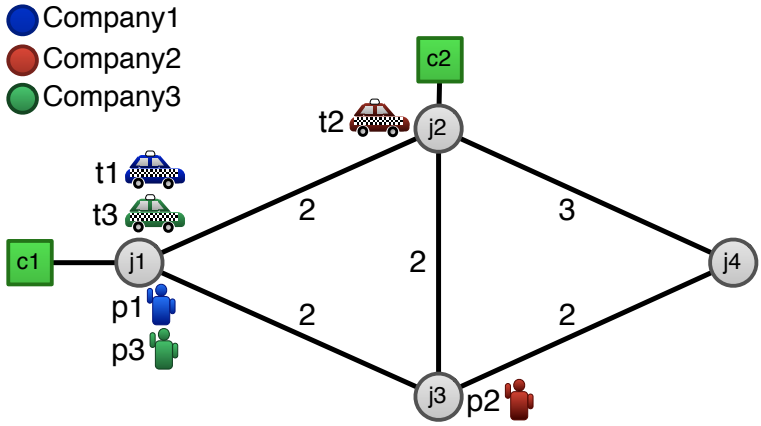


Figure 6.7: EAV problem example representation

In this problem, the cost of an individual plan, $costP(\pi^i)$, is obtained as the sum of the costs of the actions in π^i. We assume unitary costs for all actions except for the `drive` actions, whose cost depends on the length of the street, as shown in the edges of Figure 6.7. The cost of integrating a plan in a joint plan, $costS(\pi^i, \Pi^{-i})$, includes the cost of possible delays to avoid conflicts and congestion. The cost of a delay is measured as the difference in the number of time steps between the finish time of π^i in isolation and when π^i is integrated

in Π^{-i} multiplied by a constant. This constant depends on the impact of a delay on each agent, which in turn may depend on whether or not a passenger is waiting for the taxi. For the sake of simplicity, we will assume a constant value of 5 units to all agents. The cost of a congestion is linear with the number of congested actions returned by the function $\#(\Pi, t, r)$, for any agent i and resource r; i.e., if two actions use the same resource simultaneously, the involved agents get a cost rise of 2; if three actions are involved, then the cost rise is 3, and so on. Additionally, we set $cc^i = 10000$ to obtain the value of $costU(\pi^i, \Pi^{-i})$. Despite the above specifications, we note that the IPG cost functions can be individually customized to each agent accordingly to its preferences.

Table 6.4: Individual agents' plans

$time$	Company1 (π_1^1)	Company2 (π_1^2)	Company3 (π_1^3)
0	charge t1 j1 c1 n1 l0 l2	charge t2 j2 c2 n1 l0 l2	charge t3 j1 c1 n1 l0 l2
1	leave-charger t1 c1 j1	leave-charger t2 c2 j2	leave-charger t3 c1 j1
2	pick-up-passenger t1 p1 j1	drive t2 j2 j3 l2 l1	pick-up-passenger t3 p3 j1
3	drive t1 j1 j3 l2 l1	pick-up-passenger t2 p2 j3	drive t3 j1 j3 l2 l1
4	drive t1 j3 j4 l1 l0	drive t2 j3 j4 l1 l0	drive t1 j3 j4 l1 l0
5	drop-passenger t1 p1 j4	drop-passenger t2 p2 j4	drop-passenger t3 p3 j4
-	$costP(\pi_1^1) = 8$	$costP(\pi_1^2) = 8$	$costP(\pi_1^3) = 8$

Table 6.4 shows the best individual plan of each company. The goal of Company1 and Company3 is to carry a passenger (p1 and p3, respectively) from j1 to j4, while the goal of Company2 is to take p2 from j3 to j4. The costs of these optimal plans are: $costP(\pi_1^1) = costP(\pi_1^2) = costP(\pi_1^3) = 8$. We will compare these plans, which maximize the individual utility (minimize the cost) of each company agent, to the final plans of the solution joint plan.

As explained in Section 5.3, an order between the agents is established. We will assume Company1 goes first, followed by Company2 and then Company3.

The initial joint plan is built in the first iteration of BRPS, starting from $\Pi = \varnothing$, and no upper cost bound for any agent.

- **Iteration 1**:
  - Company1 generates its plan π_1^1 with $costTotal(\pi_1^1, \Pi^{-1}) = 8$ (see Table 6.4). The current joint plan is $\Pi = \langle \pi_1^1, \varnothing, \varnothing \rangle$.
  - Company2 puts forward π_1^2 and integrates it in Π, which causes two congestion interactions. There is an electricity network congestion at $t = 0$ since t1 and t2 are using chargers c1 and c2, which are both connected to the same electricity network n1. Moreover, a traffic jam congestion arises at $t = 4$ since both taxis use the road from j3 to j4. Solving a congestion entails a delay of one time step in the finishing time of the agent multiplied by 5. If Company2 solves the congestion at $t = 0$ with one time-step delay, it will be also solving the congestion at $t = 4$ since the whole plan is delayed one time unit. Then, solving the two congestion interactions is a total cost of 5. However, remaining in congestion (cost rise of 2 per congestion) is less costly for Company2 than solving the two congestion interactions. Thus, the cost of integrating π_1^2 in Π^{-2} is the sum of the individual plan cost plus the congestion cost; that is, $costTotal(\pi_1^2, \Pi^{-2}) = 8 + 2 + 2 = 12$. The resulting joint plan is $\Pi = \langle \pi_1^1, \pi_1^2, \varnothing \rangle$.
  - Company3 integrates π_1^3 in Π and finds out that t3 causes a conflict to t1 due to the simultaneous use of c1. Company3 addresses the conflict through an inter-agent ordering that delays the execution of its plan two time steps. This outcome is preferable for Company3, because being in a planning conflict would report it a significantly

higher cost. Therefore, the cost for Company3 is the sum of the cost of π_1^3 plus the delay cost, $costTotal(\pi_1^3, \Pi^{-3}) = 8 + 2 * 5 = 18$. At this point: $\Pi = \langle \pi_1^1, \pi_1^2, \pi_1^3 \rangle$.

- **Iteration 2**:

  - Company1 examines the cost of π_1^1 in Π and finds out that it is higher than expected due to the two congestions with Company2; that is, $costTotal(\pi_1^1, \Pi^{-1}) = 8 + 2 + 2 = 12$. Subsequently, Company1 runs the search procedure with an upper cost bound $upper^1 = 12$, synthesizing π_2^1, a plan that traverses the street between j2 and j4. This plan is a better response because $costTotal(\pi_2^1, \Pi^{-1}) = 9 + 2 = 11$. Despite the fact that traversing the street j2-j4 is more costly than j3-j4, π_2^1 allows Company1 to avoid the congestion in j3-j4, which results in a better-cost outcome. We note that t1 does not avoid the electricity network congestion with t2 because it is unable to do so. Then, the resulting joint plan is $\Pi = \langle \pi_2^1, \pi_1^2, \pi_1^3 \rangle$.

  - Company2 examines the cost of its plan π_1^2, $costTotal(\pi_1^2, \Pi^{-2}) = 8 + 2 = 10$. The cost of π_1^2 is reduced thanks to the introduction of π_2^1 by Company1, which addresses a congestion that affected Company1 and Company2, thus benefiting both agents. Company2 executes the search process with $upper^2 = 10$ and it does not find a better response after exhausting the search space. Therefore, Company2 maintains its initial plan π_1^2 and the joint plan remains unchanged, $\Pi = \langle \pi_2^1, \pi_1^2, \pi_1^3 \rangle$.

  - Company3 analyzes its plan, which has the same cost as in the previous iteration, $costTotal(\pi_1^3, \Pi^{-3}) = 8 + 2 * 5 = 18$. Company3 is un-

able to obtain a better response, and thus, it maintains π_1^3. Hence, $\Pi = \langle \pi_2^1, \pi_1^2, \pi_1^3 \rangle$.

- **Iteration 3**:

  - Company1 checks the cost of its plan, $costTotal(\pi_2^1, \Pi^{-1}) = 9+2 = 11$, and it does not find a better plan after searching. Since Company1 does not changes its plan, either will Company2 and Company3. Given that no agent changed its plan in a complete iteration, BRPS converges to the current joint plan Π, which is an IPG solution.

Table 6.5: Resulting IPG solution joint plan Π

$time$	Company1 (π_2^1)	Company2 (π_1^2)	Company3 (π_1^3)
0	*charge t1 c1 n1 l0 l2*	*charge t2 c2 n1 l0 l2*	-
1	leave-charger t1 c1 j1	leave-charger t2 c2 j2	-
2	pick-up-passenger t1 p1 j1	drive t2 j2 j3 l2 l1	charge t3 c1 n1 l0 l2
3	drive t1 j1 j2 l2 l1	pick-up-passenger t2 p2 j3	leave-charger t3 c1 j1
4	drive t1 j2 j4 l1 l0	drive t2 j3 j4 l1 l0	pick-up-passenger t3 p3 j1
5	drop-passenger t1 p1 j4	drop-passenger t2 p2 j4	drive t3 j1 j3 l2 l1
6	-	-	drive t1 j3 j4 l1 l0
7	-	-	drop-passenger t3 p3 j4
-	$costTotal(\pi_2^1, \Pi^{-1}) = 9+2 = 11$	$costTotal(\pi_1^2, \Pi^{-2}) = 8+2 = 10$	$costTotal(\pi_1^3, \Pi^{-3}) = 8+2*5 = 18$

Table 6.5 shows the final plans of the three agents in the joint plan Π. The electricity network congestion at $t = 0$ is shown in italics. In the IPG solution, the plan of Company1 is 3 units more costly than its initial individual plan due to the electricity network congestion, and also because it changed its initial route and switched to a different plan. Company2 also experienced a cost rise of 2 units due to the congestion with Company1. Finally, the plan of Company3 is 10 units more costly than its best individual plan because of a delay of two time steps that avoids a conflict with Company1. This coordinated solution

satisfies all agents since they are in a PNE, and thus, any unilateral deviation will jeopardize the execution of their plans.

We must note that a different order of the agents, for instance if `Company3` was ordered before `Company1`, would give rise to a different solution joint plan because `Company3` would be the first to occupy the charger `c1`.

6.4.2 Results

In this section, we perform an empirical evaluation of the defined EAV domain over a synthetic benchmark of problems. On the one hand, we make a general analysis of the performance and solution quality of our BRPS approach presented in Chapter 5. For this analysis, we measure the total computation time and the global cost (sum of all agents' cost) of an IPG solution with our BRPS approach against the BRP approach of (Jonsson et al. 2011). On the other hand, we perform an analysis of the results from the individual agents' point of view. In this case, we analyze the strategic behavior adopted by the BRPS agents depending on the configuration of their cost functions, and we also analyze how the order of the agents in the better-response dynamics of BRPS influences the results.

Comparative Evaluation of BRPS and BRP

In order to provide a general analysis of the performance of BRPS and BRP at solving our EAV domain, we prepared a synthetic benchmark that includes 25 multi-agent problems of growing complexity. Table 6.6 shows the problem setup of this benchmark. The columns of Table 6.6 indicate the number of *company agents*, *taxis* and *passengers* per company, as well as the number of *junctions* and *chargers*, and the *battery capacity* of the taxis.

As shown in Table 6.6, the number of company agents per problem ranges between 2 and 6: the first 5 problems, p1-2 to p5-2, include two different agents; problems p6-3 to p10-3 feature 3 agents, and so on. In each 5-problem block, the different parameters of the task are adjusted to progressively increase the difficulty of the problems. For example, p1-2 includes 2 taxis, 2 passengers per agent, and 4 junctions, while p5-2 presents 4 taxis and 5 passengers per agent, as well as a much larger street map of 12 junctions. Other key parameters of the domain, such as the number of chargers and maximum battery capacity of the taxis, are scaled up along with the number of junctions.

The experimental results for both approaches are summarized in Table 6.7[5]. The first three columns of each planner refer to the number of actions, makespan (finish time), and cost of the solution joint plans. The next two columns show the number of iterations and computation time required by each approach to synthesize the solution joint plans. The dagger symbol (†) indicates that a solution was not found within the given time limit. The cost values used in the function $costTotal$ of BRPS are the values shown in the example of Subsection 6.4.1. Similarly, BRP was configured to apply the same costs values as BRPS, except for the cost of unsolved conflicts ($costU$), which is ignored in BRP as it always works with a conflict-free joint plan.

The computation time of the problems in Table 6.7 are mainly determined by the complexity of the street map, the number of taxis and task goals (passengers to transport) per agent. This can be observed in each block of tasks, where the resolution of a problem is generally more time-consuming than the previous problems of the block. The computation time grows exponentially in the last problems of each block as they represent the most complex maps in the number of junctions, taxis and passengers. For this reason, convergence

[5] All the tests were conducted on a single machine with an Intel Core i7-3770 CPU at 3.40GHz and 8 GB RAM. Each test was run within a time limit of 1800 seconds.

Table 6.6: Problem setup of the benchmark of tests for the EAV domain

Prob-Ag	Companies	Taxis	Passengers	Junctions	Chargers	Battery
p1-2	2	2	2	4	1	4
p2-2	2	2	3	6	2	6
p3-2	2	3	3	8	2	8
p4-2	2	3	4	10	3	10
p5-2	2	4	5	12	3	12
p6-3	3	2	2	4	1	4
p7-3	3	2	3	6	2	6
p8-3	3	2	4	6	2	6
p9-3	3	3	3	8	2	8
p10-3	3	3	4	10	3	10
p11-4	4	2	2	4	1	4
p12-4	4	2	3	6	2	6
p13-4	4	2	4	6	2	6
p14-4	4	2	3	8	2	8
p15-4	4	3	3	8	2	8
p16-5	5	2	2	4	1	4
p17-5	5	2	3	6	2	6
p18-5	5	2	4	6	2	6
p19-5	5	2	3	8	2	8
p20-5	5	3	3	8	2	8
p21-6	6	2	2	4	1	4
p22-6	6	2	3	6	2	6
p23-6	6	2	4	6	2	6
p24-6	6	2	3	8	2	8
p25-6	6	3	3	8	2	8

to an IPG solution requires significantly larger computation times in these problems.

Table 6.7: Experimental results of BRP and BRPS in the EAV domain

	BRP					BRPS				
Prob-Ag	Acts	Ms	Cost	Iters	Time	Acts	Ms	Cost	Iters	Time
p1-2	16	9	22	2	2.84	16	9	22	2	**0.66**
p2-2	23	13	49	**2**	38.29	23	**10**	**34**	3	**35.58**
p3-2					†	25	8	36	2	286.99
p4-2					†	37	12	49	2	483.67
p5-2					†	41	8	54	3	954.38
p6-3	**26**	**10**	40	**2**	5.42	27	11	38	3	**1.79**
p7-3	40	18	93	2	408.68	40	**11**	**58**	2	**31.87**
p8-3					†	48	16	66	3	239.06
p9-3					†	39	6	58	2	223.17
p10-3					†	48	14	67	3	749.68
p11-4	**37**	12	84	**2**	14.63	41	**10**	**72**	3	**5.01**
p12-4					†	54	15	78	3	118.83
p13-4					†	57	12	80	3	439.04
p14-4					†	54	12	80	3	658.39
p15-4					†	50	11	74	3	1052.07
p16-5	43	14	78	**2**	24.32	43	14	78	3	**5.38**
p17-5					†	74	17	110	2	278.69
p18-5					†	68	16	94	3	251.46
p19-5					†	62	12	94	3	222.00
p20-5					†	64	11	96	2	1167.65
p21-6					†	61	13	100	4	29.93
p22-6					†	71	11	106	3	202.26
p23-6					†	87	14	122	3	1665.96
p24-6					†	80	15	118	3	1761.50
p25-6					†	72	12	108	3	1643.19

Despite the complexity of some of the problems, our BRPS approach solves the complete benchmark, generating solution plans of up to 87 actions. BRP,

however, is only able to solve 6 problems within the time limit, being unable to attain any problem of the fifth block. In summary, BRPS reaches 100% coverage, while BRP only solves 24% of the benchmark problems, which proves that our approach scales up significantly better than BRP.

Regarding computation time, BRPS is in general one order of magnitude faster than BRP, with the only exception of problem p6-3. We must further note that the results of BRP in Table 6.7 do not reflect the time needed for the calculation of the initial conflict-free joint plan, a time-consuming task that is not required in BRPS. All in all, we can conclude that BRPS clearly outperforms BRP in terms of computation time.

BRP only needs 2 iterations to converge to a solution in 6 of the problems, while BRPS takes one more iteration in some of these problems. This is explained because starting the search process from a conflict-free plan facilitates reaching a solution, while BRPS needs to run as many iterations as number of agents to build the first joint plan. Additionally, better-response dynamics may take more iterations to converge since agents do not necessarily propose the best possible response at each iteration. Despite the downside to a slow convergence, BRPS exhibits a significantly shorter computation time per iteration than BRP, which results in a superior performance and scalability. BRPS was able to converge to a solution in all the problems of the benchmark since better-response dynamics rarely get into a cycle, as pointed out in Section 5.3.3.

BRPS proves to be particularly efficient at optimizing the makespan of the solution joint plans. Even though many of the solution plans contain a large number of actions, the makespan of such plans never exceeds 17 time units, which proves that our approach excels at enforcing parallelism among the company agents' actions. In other words, the company agents in BRPS use

their available taxis in a concurrent and efficient manner, effectively minimizing makespan and cost. This is also supported by the partial-order reasoning mechanism of the planner MH-FMAP (Torreño, Sapena, et al. 2015). In contrast, the BRP plans finish later because the planner used by a `company` agent to calculate a plan does not parallelize the actions of its taxis. This has also a direct impact in the cost of the joint plans of BRP, making the execution of such plans require more time steps.

In general, almost all BRP plans have a significantly higher cost than the solutions of BRPS. For example, in problem p2-2, BRP obtains a solution joint plan of 49 cost units, while BRPS yields an IPG solution of 34 cost units. Similarly, the cost of BRP for the problem p7-3 is 93 compared to the 58 cost units of BRPS. Again, these differences are mainly due to the fact that the type of planner used by the agents in BRP does not enable parallelizing the plan actions; this results in a later makespan, which in turn penalizes the cost of the solution plans.

We can also observe in Table 6.7 that the cost values and number of actions do not generally scale up and this is specially notable in problems that feature similar cost values but a significantly different number of actions; e.g., the solution plan to problem p21-6 has 61 actions and 100-unit cost whereas the solution to p25-6 has 72 actions and 108-unit cost. Aside from the fact that the cost of the `drive` actions range between 2 and 3 units, unlike the rest of actions that have unitary costs, problems like p21-6 occur in smaller size cities (fewer junctions and chargers) and so it is more likely to have congestion and conflict interactions. Consequently, problems that happen is smaller cities tend to have a relatively higher cost due to the more frequent appearance of congestions and the introduction of delays to avoid congestions or battery charging conflicts.

In summary, despite the notable complexity of the EAV benchmark, which results in solution plans of more than 40 actions in most cases, our BRPS approach exhibits an excellent behavior, outperforming BRP in all the evaluated metrics:

1. **Coverage**: BRPS solves the complete benchmark within the given time limit, while BRP only solves 6 of the simplest problems of the benchmark (24% coverage).

2. **Execution time**: Despite better-response converge more slowly than best-response, BRPS is one order of magnitude faster than BRP in almost all cases.

3. **Makespan**: The underlying MAP machinery of BRPS efficiently enforces parallelism among the agents' actions, keeping the makespan of the solution plans below 18 time units in all cases.

4. **Cost**: The cost of the solutions plans is significantly lower in BRPS than in BRP. The lack of parallelism in the BRP penalizes the plan cost notably, while our approach ensures the generation of robust parallel plans where the taxis of a company agent act in parallel whenever possible.

All in all, the experimental results prove that BRPS significantly outperforms the state-of-the-art BRP approach in both sheer performance and plan quality, thus emerging as the current top-contending technique in non-cooperative MAP.

Analysis of the Strategic Behavior of the BRPS Agents

This section analyzes the strategic behavior adopted by the BRPS agents accordingly to the configuration of their cost functions. We do not present a comparative evaluation with the strategic behavior of agents in BRP because the cooperative nature of the initial joint plan of BRP would render the comparison not meaningful. More specifically, the behavior of agents in BRP, which must best respond to a cooperative solution by maintaining the conflict-free structure of the plan, limit the choice of action of the agents as to satisfying their own private interests.

For this analysis, we used the problem example presented in Section 6.4.1 and depicted in Figure 6.7 except that the battery level of the taxis is 1 (`level l1`). The default ordering of the agents during the better-response dynamics of BRPS is `Company1-Company2-Company3`. We tested this problem in six different settings that modify the agents' cost functions. The columns of Table 6.8 show the number of actions, makespan, and cost of the plans of each `company` agent in the six different settings.

Table 6.8: Strategic behavior analysis for different cost functions

	Company1			Company2			Company3		
Setting	Act	Ms	Cost	Act	Ms	Cost	Act	Ms	Cost
1	6	6	11	6	6	10	6	8	18
2	6	6	11	6	6	10	6	8	10
3	6	6	11	6	6	10	6	8	68
4	6	6	11	6	6	18	6	8	19
5	6	6	8	7	8	15	6	10	28
6	6	10	28	7	8	15	6	6	8

In the following, we analyze the six configurations used in this experiment and the results summarized in Table 6.8:

- **Setting 1**: This is the **original setting** of the problem as presented in the example of Figure 6.7, where the **cost of a delay of one time step is 5**. The cost of `drive` actions is defined as the length of the street they traverse, which is 2 by default, except for the street `j2-j4`, whose length is 3 (see Figure 6.7). The rest of actions have unitary cost. Congestions cause a cost increase which is linear with the number of congested agents; that is, given a congestion that affects two agents, the cost of their actions is increased by 2 units; if there are three agents in the congestion, the congestion cost is 3, and so on.

  The solution plan for this setting is shown in Table 6.5. As explained in the example of Figure 6.7, there is an electricity network congestion between `Company1` and `Company2` because they are using two chargers connected to the same electricity network at `t=0`. `Company3` is delayed two time steps because it must wait for the charger `c1` to be released released by `Company1`.

- **Setting 2**: In this setting, the **cost of a delay of one time step is 1**. The rest of the costs are as in setting 1. The solution joint plan for this setting is the same as in setting 1. However, the plan of `Company3` has a lower cost (10 cost units) because it benefits from the unitary delay cost.

- **Setting 3**: In this case, the **cost of a delay of one time step is 30** for the three agents, while the rest of costs remain as in setting 1. Again, the only affected agent is `Company3`, which does not change its plan, but reports a total cost of 68 units because of the higher cost of a delay.

- **Setting 4**: In this configuration, we defined a **10-unit cost for driving through street `j3-j4`**, keeping the rest of costs unaltered with respect to setting 1. The plan of `Company2` still uses street `j3-j4` to take the passenger to the goal destination. The best solution for `Company2` would be to take a longer path through streets `j3-j2` and `j2-j4` because this would report a lower cost than using street `j3-j4`. However, since it is not possible to charge the battery with a passenger on the taxi, and the maximum capacity of the battery is limited to 2 units, `Company2` cannot take this alternative route. Regarding `Company3`, its taxi waits for 2 time steps until `taxi1` finishes charging the battery at `c1`, and then, it takes the paths `j1-j2` and `j2-j4` to avoid the more costly street `j3-j4`.

- **Setting 5**: This setting **increases the congestion cost** as follows: a 2-agent congestion reports the involved agents a 12-unit cost; a 3-agent congestion entails a 13-unit cost, and so on. The rest of costs remain as in setting 1. The IPG solution obtained with this setting is shown in Table 6.9. This joint plan presents several differences with respect to the solution of setting 1 that concern `Company2` and `Company3`. In this solution, `taxi2` of `Company2` drives from `j2` to `j1` to charge its battery at t=2, once `taxi1` leaves charger `c1`. This explains the 15-unit cost reported by `Company2`, which is slightly higher than `Company1`'s cost. `Company2` makes this decision to prevent `taxi1` and `taxi2` from charging their batteries simultaneously at chargers `c1` and `c2`, which would cause a network congestion. Then, `taxi2` drives to `j3` to pick up its endowed passenger and transports him to `j4`. Therefore, the makespan and cost increase reported by `Company2` in this setting is explained by the additional action that drives `taxi2` to charger `c1`, and the subsequent 1-time-step delay. Finally, `Company3` reports a higher cost than the rest of agents (28 units), because it waits for 4 time steps until the charger `c1` becomes available.

The high cost of congestions in this setting forces the agents to introduce delays to charge the batteries of their taxis in a sequential order. Consequently, the agent that revises first its plan in the first iteration (`Company1`) is favored since the best option for the subsequent agents is to delay their activities until the first agent releases a key resource (in this case, the charger `c1`).

Table 6.9: Resulting IPG solution joint plan Π for setting 5

t	`Company1` (π_2^1)	`Company2` (π_1^2)	`Company3` (π_1^3)
0	`charge t1 c1 n1 l1 l2`	`drive t2 j2 j1 l1 l0`	-
1	`leave-charger t1 c1 j1`	-	-
2	`pick-up-passenger t1 p1 j1`	`charge t2 c1 n1 l0 l2`	-
3	`drive t1 j1 j3 l2 l1`	`leave-charger t2 c1 j1`	-
4	`drive t1 j3 j4 l1 l0`	`drive t2 j1 j3 l2 l1`	`charge t3 c1 n1 l1 l2`
5	`drop-passenger t1 p1 j4`	`pick-up-passenger t2 p2 j3`	`leave-charger t3 c1 j1`
6	-	`drive t2 j3 j4 l1 l0`	`pick-up-passenger t3 p3 j1`
7	-	`drop-passenger t2 p2 j4`	`drive t3 j1 j3 l2 l1`
8	-	-	`drive t1 j3 j4 l1 l0`
9	-	-	`drop-passenger t3 p3 j4`
-	$costTotal(\pi_1^1, \Pi^{-1}) = 8$	$costTotal(\pi_3^2, \Pi^{-2}) = 10 + 1 * 5 = 15$	$costTotal(\pi_1^3, \Pi^{-3}) = 8 + 4 * 5 = 28$

- **Setting 6**: This setting maintains the costs of setting 5, but the **ordering of the agents in BRPS is reversed**; that is, `Company3` goes first, followed by `Company2` and `Company1`. As expected, the results are also reversed with respect to setting 5: in this case, `Company3` presents no delay in its execution while `Company1` does. `Company2` keeps the same solution plan and cost as in setting 5.

In these experiments, we can observe that agents design their strategies (plans) to optimize cost according to the specification of their cost functions. Agents try to find the lowest-cost plan taking into account their own cost functions and the plans of the other agents. Moreover, agents avoid the most costly situations if they are able to do so. For instance, if remaining in a congestion

entails a cost higher than escaping from it by delaying actions, agents will opt for delaying the execution of their actions. All in all, we can conclude that, as expectedly, agents in BRPS follow a strategic behavior regarding their cost functions.

Influence of the Ordering of BRPS Agents

In this section, we analyze whether the order of agents in BRPS affect the cost of an agent's plan with respect to its best individual plan as a stand-alone agent.

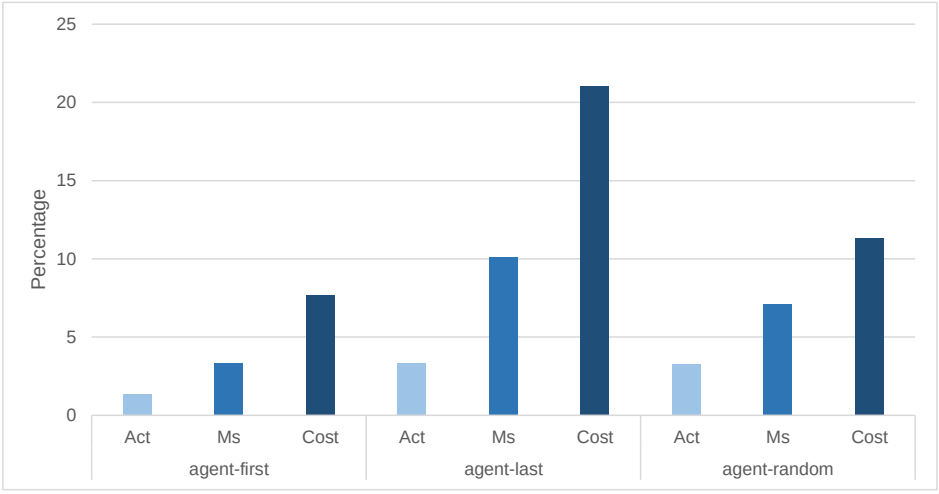


Figure 6.8: Average increment in percentage of actions, makespan and cost of one agent in the IPG solution with respect to its best individual plan, when it goes first, last, or random in the BRPS order

Figure 6.8 shows the results when an agent is the first one (*agent-first*) in the arbitrary order of the BRPS process, when it is the last one (*agent-last*) or when its position is randomly chosen (*agent-random*). We show the average increment in the number of actions, makespan and cost of a specific agent for

all problems of Table 6.7 with respect to the best individual plan of the agent, which is computed exhausting the search space of the agent.

According to Figure 6.8, we can observe that the order of the agent has a significant impact in the results. The best results are for *agent-first* since this is the agent that reaches first the charger, compelling the other agents to use alternative plans or introduce a delay. This is also reflected in the number of actions of *agent-first*, which only increases 1.33% with respect to the number of actions of its best individual plan. On the other hand, the makespan increases slightly and the cost is 7.65% higher because of the unavoidable congestions. In the case of *agent-last* or *agent-random*, the number of actions only increases 3.3% while the makespan and cost rise notably (21% increase in the cost of *agent-last*). The difference between *agent-first* and *agent-last* lies in the number of conflicts the agent needs to solve delaying its execution.

We can conclude that the arbitrary order of the agents clearly impacts the results of BRPS. The first agent is clearly favored over the others, while a random order seems a fairer option. Another interpretation within an arbitrary order in a blackboard system is that the agent that communicates first its plan is in a more advantageous position. Nonetheless, BRPS is designed to solve problem sets rather than a single problem. Thereby, selecting a random order in each problem would balance the agents' costs across the whole problem set.

6.5 General Discussion on the Results

This chapter experimentally compared the performance of BRPS against BRP, one of the few available state-of-the-art approaches to non-cooperative MAP. The tests performed along this chapter are organized in three different scenarios that feature congestions, planning conflicts, and a combination of both.

The first scenario, based on a network routing domain (Jonsson et al. 2011), puts our approach to test in a context where congestion issues may emerge among the agents. In this scenario, BRPS performs better than BRP, obtaining plans of similar quality and requiring significantly lower execution times. These results prove that, in practice, the better-response dynamics of BRPS are as effective as the best-response-based mechanism of BRP when dealing with congestion issues. Regarding execution time, BRPS is proven to be much more efficient than BRP, particularly in complex problems that feature a large number of agents or complex networks with many nodes and interconnections.

In the conflict-based scenario, which includes problems of several CoDMAP domains adapted to a non-cooperative planning context, BRPS is again the superior approach, since in general, it yields better solution joint plans and converges much faster than BRP.

In terms of plan quality, BRPS outperforms BRP in *rovers* problems and obtains similar plans in the rest of domains. Despite the fact that BRP departs from an initial conflict-free plan, it does not achieve better results than BRPS. The reason behind this result is that agents in BRPS explicitly use information regarding planning conflicts in their utility functions, which benefits BRPS in this scenario, easing convergence and improving the overall solution quality.

Regarding execution time, BRPS is clearly faster than BRP in this conflict-based scenario. This is particularly notable in the most complex problems of the *zenotravel* domain. Despite requiring more iterations to converge, BRPS spends much lower time per iteration than BRP.

The final scenario is based on an Electric Autonomous Vehicles (EAV) case study. This domain combines conflicts and congestions, thus giving rise to highly complex and demanding problems, as it can be noted in the high execution times of BRPS. In contrast to the previous scenarios, an agent in this

domain is a company that manages a fleet of taxis which, ideally, act in parallel. Hence, the individual plan of an agent must include parallel actions in order to maximize its quality. Taxi company agents must avoid conflicts and congestion issues to optimize traffic density and increase their profit by means of coordination.

In this EAV domain, BRPS proves to be the superior approach, outperforming BRP in all the magnitudes we measured. Despite departing from a conflict-free joint plan, BRP cannot converge to a solution in most problems (24% coverage). In contrast, our BRPS approach, which starts its better-response dynamics from scratch, obtains 100% coverage, converging in less than 30 minutes in all cases. BRPS yields plans of higher quality than those of BRP, which proves that BRPS agents efficiently parallelize the actions of their taxis while circumventing congestion issues and conflicts. The costs of the plans synthesized by BRPS are close to their number of actions, which shows that BRPS agents manage to avoid most congestions and delays.

The key advantages of BRPS over BRP that explain the previous results can be summarized as follows:

- **Better-response dynamics**: BRPS allows agents to synthesize better responses (non-optimal plans) computable in polynomial time in many domains (i.e., transport domains without fuel restrictions). In contrast, BRP adopts a best-response approach that is always NP-complete (Helmert 2003). Hence, better-response dynamics in a planning setting are faster than best-response dynamics, while obtaining solutions of a similar quality.

- **Joint plan synthesis from scratch**: while BRP requires an initial conflict-free joint plan computed by a MAP planner as an input, the BRPS approach synthesizes joint plans from scratch. The initial joint plan of

BRP, which does not reflect the agents' self interest (since it is computed by a cooperative MAP planner), is used as a reference throughout the best-response dynamics and ultimately determines the behavior of the BRP agents. Moreover, this initial joint plan is already a solution for the problem, which reduces BRP to a procedure that introduces small improvements over the initial solution. In contrast, agents in BRPS are free to compute and progressively balance a solution joint plan that meets their interests without any initial commitment or limitation.

- **Fully-distributed MAP machinery**: the use of an underlying multi-agent system ensures the **preservation of the agents' privacy**, since BRPS does not depend on a centralized planning entity that has complete access to the problem. Guaranteeing the agents' privacy is critical in a context where agents are self-interested. Hence, BRPS is a more realistic non-cooperative MAP approach than BRP.

  Additionally, the IPG does **not allow synergies** among the agents; that is, agents do not behave cooperatively since none of them can benefit from the effects of other agents' actions to achieve their goals. If synergies were allowed, as in BRP, agents would not be able to change their plans in case they were providing preconditions to other agents, because that would cause inconsistencies or conflicts. This would limit the strategic behavior of the agents, thus harming their self-interest.

- **Partial-Order Planning (POP) technology**: the POP-based planner integrated in the BRPS agents allows them to efficiently parallelize their individual plans, which is very useful in domains such as the EAV, where an agent must plan the concurrent actions of several taxis.

- **Conflicts as part of the agents' utility functions**: agents in BRPS explicitly reason about planning conflicts, which improves the efficiency of BRPS in domains where these issues can emerge.

All in all, we can conclude that BRPS is the current top-performing non-cooperative MAP approach, clearly surpassing the state-of-the-art method BRP.

6.5.1 Limitations of the Model

Despite the remarkable performance of BRPS, it is worth noting that presents some limitations:

- The outcome of the better-response dynamics depends on the order of the agents in the game. For instance, the first agent that puts forward a plan in the better-response dynamics has a competitive advantage, since it does not need to make changes that compromise the cost of its proposal in order to integrate it in the (initially empty) joint plan.
- In the context of better-response dynamics, an agent is not forced to respond with its best possible plan; this fact may affect the number of iterations required by BRPS to converge. The experimental results showed that BRPS took more iterations to converge than the best-response approach BRP in some instances. However, the execution times were generally low thanks to the reduced time per iteration exhibited by BRPS.
- Whereas BRPS converged to conflict-free solutions in all the tested problems, our approach may converge to an unfeasible joint plan in case of a multi-symmetric unsolvable situation.

The computational complexity of non-cooperative MAP task is hard in most cases, especially in domains where individual plans cannot be computed in

polynomial time. This is an inherent limitation to the type of non-cooperative MAP problems that BRPS aims to solve. In practice, BRPS solved tasks by computing joint plans up to 87 actions in less than 30 minutes, which is a remarkable size from a planning perspective and considering the difficulty of non-cooperative MAP. Nevertheless, BRPS could be further improved to attain larger tasks.

We believe that using a different machine for each agent would help to increase the performance of BRPS. This should be combined with a continuous individual exploration of the search space of each agent, which would produce alternative plans while waiting its turn in the better-response process.

The search algorithms and heuristics of BRPS agents could also be improved by using a more informed heuristic. This would help to find better solutions faster and to prune more the search space, thus producing the best plans of each agent with less exploration.

Chapter 7

Conclusions and Future Work

As stated in the Introduction of this PhD thesis, the main goal of this work is the design and development of non-cooperative Multi-Agent Planning (MAP) models to solve planning problems that involve several self-interested agents which interact in a shared environment. More precisely, we designed the FENOCOP model, where agents have a limited set of precomputed plans, and BRPS, where each agent is equipped with an embedded planner to dynamically compute a potentially unlimited set of plans (neither initially limited nor infinite). In the following, we outline the main contributions of this PhD thesis, which allowed us to attain the main objectives of this work:

- We thoroughly analyzed the state of the art in MAP with self-interested agents, confirming the absence of computational approaches that tackle the non-cooperative MAP problem. In this setting, several non-strictly competitive agents with complementary interests combine their individual plans into a stable joint solution which ensures that all plans are executable and the private interests of the participants are kept. Al-

beit many real-world scenarios feature problems that involve the coordination of multiple self-interested planning agents, the topic of non-cooperative MAP has been traditionally neglected by the MAP community. The aim of this PhD thesis is precisely to contribute to the state of the art in non-cooperative MAP by filling this gap.

- We defined FENOCOP, our initial non-cooperative MAP model for self-interested planning agents. In this approach, agents have a limited set of precomputed plans and want to execute one of their plans in a shared environment. FENOCOP is a two-stage model that allows agents to decide which plan to execute and how to schedule the execution of the actions of such plan depending on the decisions of the rest of agents. The General Game allows each agent to decide which course of action to take from its precomputed set of plans. In the Scheduling Game, agents decide how to schedule each combination of plans (one per agent) introducing empty actions to avoid conflicts. The solutions to the Scheduling Game are Nash Equilibria, Pareto optimal, and fair; that is, they are stable solutions that satisfy agents as much as possible.

  FENOCOP contributes to the state of the art in non-cooperative MAP in several ways. Firstly, the outcome of this computational model is a stable joint plan where all the agents solve their planning tasks. Secondly, the calculation of multiple Nash Equilibria allows for the selection of solutions that are also Pareto optimal and fair. These properties guarantee that no agent can improve its utility without reducing the utility of other agents, and that the less satisfied agent is as satisfied as possible (egalitarian social welfare).

- The second model defined in the context of this PhD thesis, named BRPS, is an approach based on better-response dynamics that iteratively con-

verges to a stable joint plan. In BRPS, each agent has the ability to synthesize and propose a potentially unlimited amount of plans (not infinite since it is bounded by plans cost and the search space of the agent) during the better-response dynamics.

Among the contributions of BRPS, we can highlight the fact that agents make use of individual cost functions that evaluate the quality of their plans according to their inherent cost and the interactions with other agents; i.e., planning conflicts and congestions. Agents do not depend on a centralized entity and they iteratively revise and combine their plans in order to come up with a conflict-free solution joint plan. Additionally, the combination of individual plans cost, planning conflicts, and congestion in the agents' utility functions makes BRPS a realistic model that reflects the actual behavior of real-world agents.

While under some conditions convergence to a Nash equilibrium cannot be guaranteed in BRPS, the empirical results of our approach show that better-response dynamics converge to stable solutions in most instances. Finally, BRPS is computationally efficient since better-response dynamics in MAP problems are computable in polynomial time under some assumptions.

- Both theoretical models were fully implemented and empirically evaluated. This is an important contribution since most game-theoretic models are not typically implemented and tested exhaustively. The results of FENOCOP show that the Scheduling Game obtains better quality results when Pareto optimality and fairness are considered against algorithms that only guarantee equilibrium solutions. However, these experiments also showed that FENOCOP could be computationally intractable when the number of plans and/or agents are relatively high.

On the other hand, the BRPS approach is built upon the MAP solver MH-FMAP. We implemented the better-response dynamics mechanism as well as the underlying techniques to compute the cost of the planning conflicts, congestion issues and individual plans. The empirical evaluation of BRPS compares it against the state-of-the-art BRP approach (Jonsson et al. 2011). The experimental tests include three scenarios: a setting which features only congestions, a test based on several CoDMAP domains where conflicts among agents may emerge; and a custom case study that features both planning conflicts and congestions. BRPS clearly outperforms BRP in terms of computation time while yielding solution joint plans of generally superior quality. Despite the lack of convergence guarantees of the BRPS approach under some conditions, the results prove that, in practice, convergence is achievable in most cases.

7.1 Future Lines of Research

Due to the novelty of non-cooperative MAP, this research field offers vast opportunities for future work. The next paragraphs describe some lines of work that we consider as potentially interesting for future research:

- Design of specific heuristics for non-cooperative MAP problems. These heuristics would take into account the self-interest of the agents to predict in advance how other agents may act (their strategic behavior), thus yielding a faster convergence to equilibrium solutions. The development of non-cooperative heuristics is a completely novel research line inspired by existing cooperative MAP global heuristics, preference planning, and algorithms to compute equilibrium solutions.

- Extension of the models to temporal planning. The underlying planning techniques of our models manage time constraints to efficiently schedule and parallelize actions. However, they do not manage actions with different durations. In temporal planning, the definition of a state includes information about the current absolute time and how far the execution of each active action has proceeded. Using temporal planning in our approaches would increase the overall difficulty of the task due to the complexity of this planning paradigm, but temporal planning is more accurate than classical planning to model real-world scenarios.

- Development of a framework that combines FENOCOP and BRPS. FENOCOP would be used in specific situations with small sets of plans and a low number of agents to attain equilibria that are Pareto optimal and fair. Then, BRPS would be used as a general-purpose solver since it is capable of tackle larger problems.

- Study of other techniques applicable to the non-cooperative MAP problem. Despite the fact that agents in non-cooperative MAP have private interests, and hence, game theory is necessary to guarantee stable solutions, other techniques may be applicable. For instance, argumentation protocols could be interleaved during the planning process in order to reach agreements that must be guaranteed by external entities. We believe that it would be interesting to analyze how joint plans for self-interested agents could be built using agreement technologies.

7.2 Related Research Activities

This section lists the research activities performed during the development of this PhD thesis, namely, the related scientific publications, research stays, and research projects.

7.2.1 Related Publications

The following subsections list all the author's related scientific publications. We classify articles according to the type of publication they were included into:. The first subsection presents the articles appearing in journals listed in Science Citation Index (SCI)[1]. Then, the second subsection cites the papers published in the proceedings of relevant conferences included in the Computing Research and Education Association of Australasia (CORE)[2] rankings. Finally, the third subsection lists other relevant scientific articles without an impact factor or not published in a ranked conference.

Publications in SCI journals

- J. Jordán, A. Torreño, M. de Weerdt, and E. Onaindía. A Better-Response Strategy for Self-Interested Planning Agents. *Applied Intelligence*. 2017 *ACCEPTED MINOR CHANGES*. **Impact Factor (2016): 1,904. Q2**.

- J. Jordán, M. de Weerdt, A. Torreño, and E. Onaindía. A Non-Cooperative Game-Theoretic Approach for Conflict Resolution in Multi-Agent Planning. *Group Decision and Negotiation, Special Issue on Artificial Intelligence Techniques for Conflict Resolution.* 2017 *SUBMITTED*. **Impact Factor (2016): 1,688. Q1 (social sciences)**.

[1] http://ip-science.thomsonreuters.com/cgi-bin/jrnlst/jloptions.cgi?PC=K
[2] http://www.core.edu.au/

- J. Jordán, S. Heras, S. Valero, and V. Julian. An Infrastructure for Argumentative Agents. *Computational Intelligence*. Volume 31(3), pages 418–441, 2015. **Impact Factor (2015): 0,722. Q4**.

- S. Heras, J. Jordán, V. Botti, and V. Julian. Case-based Strategies for Argumentation Dialogues in Agent Societies. *Information Sciences*. Volume 223(20), pages 1–30, 2013. **Impact Factor (2013): 3,893. Q1**.

- S. Heras, J. Jordán, V. Botti, and V. Julian. Argue to Agree: A Case-Based Argumentation Approach. *International Journal of Approximate Reasoning*. Volume 54(1), pages 82–108, 2013. **Impact Factor (2013): 1,977. Q1**.

Publications in CORE conferences

- J. Jordán and E. Onaindía. Game-theoretic Approach for Non-Cooperative Planning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*. Pages 1357–1363, 2015. **Conference ranking: CORE A***.

- I. Sánchez-Garzón, J. Fedz-Olivares, E. Onaindía, G. Milla-Millán, J. Jordán, and P. Castejón. A multi-agent planning approach for the generation of personalized treatment plans of comorbid patients. In *14th Conference on Artificial Intelligence in Medicine*. Volume 7885, pages 23–27, 2013. **Conference ranking: CORE A**.

- A. Costa, S. Heras, J. Palanca, J. Jordán, P. Novais, and V. Julian. Argumentation Schemes for Events Suggestion in an e-Health Platform. In *XII International Conference on Persuasive Technology*. Volume 10171 of LNCS, pages 17–30, 2017. **Conference ranking: CORE B**.

- J. Jordán, S. Heras, S. Valero, and V. Julian. ArgCBR-CallCentre: A Call Centre based on CBR Argumentative Agents. In *11th Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS-13)*. Volume 7879, pages 292–295, 2013. **Conference ranking: CORE C**. (**2nd IBM prize of Scientific Excellence**).

- J. Jordán, S. Heras, and V. Julian. Case-based Argumentation Infrastructure for Agent Societies. In *7th International Conference on Hybrid Artificial Intelligence Systems (HAIS-12)*. Volume 7208(I), pages 13–24, 2012. **Conference ranking: CORE C**.

- J. Jordán, S. Heras, and V. Julian. A Customer Support Application Using Argumentation in Multi-Agent Systems. In *14th International Conference on Information Fusion*. Pages 772–778, 2011. **Conference ranking: CORE C**.

- J. Jordán, S. Heras, S. Valero, and V. Julian. An Argumentation Framework for Supporting Agreements in Agent Societies Applied to Customer Support. In *6th International Conference on Hybrid Artificial Intelligence Systems (HAIS-11)*. Volume 6678, pages 396–403, 2011. **Conference ranking: CORE C**.

Other publications

- A. Costa, S. Heras, J. Palanca, J. Jordán, P. Novais, and V. Julian. Using Argumentation Schemes for a Persuasive Cognitive Assistant System. In *4th International Conference on Agreement Technologies*. In Press. 2016.

- S. Heras, J. Jordán, V. Botti, and V. Julian. Arguing to Support Customers: the Call Centre Study Case. *Agreement Technologies*. Book chapter, pages 507–527. Springer, 2013.

- J. Jordán, S. Heras, and V. Julian. Argumentation Tool that Enables Agents to Argue. In *1st International Conference on Agreement Technologies*. Pages 455–456, 2012.

7.2.2 Scientific Research Stays

The following research stay was completed during the research period associated to this PhD thesis:

- 1-04-2015 to 15-07-2015. *Delft University of Technology, The Netherlands*. Research stay supervised by Dr. Mathijs de Weerdt in the *Algorithmics* research group of the *Software and Computer Technology* Department. Focused on game theory and developing new models for game-theoretic multi-agent planning.

7.2.3 Research Projects

This work has been performed in the context of several research projects that provided economical funding or technological support to its development:

- "**GLASS**: Goal management for Long-term Autonomy in Smart citieS" under grant MICINN TIN2014-55637-C2-1 (Main Researcher: Eva Onaindia). The main objective of the project is to analyze the problem of goal management for long-term autonomous systems, design appropriate algorithms for addressing the different components of goal management, and develop software tools that help on the application of this technology to Smart Cities tasks.

- "**PlanInteraction**: Multi-agent Interaction for Planning" under grant MICINN TIN2011-27652-C03 (Main Researcher: Eva Onaindia). This project aims to develop new agent techniques based on social dynamics for the design of a MAP platform composed of autonomous and, possibly heterogeneous, planning entities. The platform tackles aspects such as multi-agent execution, cooperative and non-cooperative MAP, plan merging and planning via argumentation.

- **"Integra - Boeing"**: Project in collaboration with Boeing Research and Technology Europe S.L. (Main Researcher: Vicente Botti).

Additionally, this work has been partly supported by the project PROMETEO II/2013/019 funded by the Valencian Government. Finally, this research would not have been possible without a 4-year FPI research scholarship TIN2011-27652-C03-01 granted to the author of this PhD thesis by the Spanish Government.

References

Aghighi, M. and C. Bäckström (2016). “A Multi-Parameter Complexity Analysis of Cost-Optimal and Net-Benefit Planning”. In: *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 2–10 (cit. on p. 116).

Applegate, C., C. Elsaesser, and J. Sanborn (1990). “An Architecture for Adversarial Planning”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 20.1, pp. 186–194 (cit. on p. 24).

Arrow, K. J. (1963). *Social Choice and Individual Values*. 12. Yale University Press (cit. on pp. 30, 34).

Bäckström, C., P. Jonsson, S. Ordyniak, and S. Szeider (2015). “A Complete Parameterized Complexity Analysis of Bounded Planning”. In: *Journal of Computer and System Sciences* 81.7, pp. 1311–1332 (cit. on p. 116).

Bäckström, C. and B. Nebel (1995). “Complexity Results for SAS+ Planning”. In: *Computational Intelligence* 11.4, pp. 625–655 (cit. on pp. 14, 115).

Banerjee, S., H. Konishi, and T. Sönmez (2001). “Core in a Simple Coalition Formation Game”. In: *Social Choice and Welfare* 18.1, pp. 135–153 (cit. on p. 23).

Barrett, A. and D. S. Weld (1994). “Partial-Order Planning: Evaluating Possible Efficiency Gains”. In: *Artificial Intelligence* 67.1, pp. 71–112 (cit. on p. 16).

Benton, J., A. Coles, and A. Coles (2012). “Temporal Planning With Preferences and Time-Dependent Continuous Costs”. In: *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 2–10 (cit. on p. 17).

Benton, J., K. Talamadupula, P. Eyerich, R. Mattmuller, and S. Kambhampati (2010). “G-Value Plateaus: A Challenge for Planning”. In: *Proceedings 20th International Conference on Automated Planning and Scheduling (ICAPS)* (cit. on p. 115).

Bercher, P. and R. Mattmüller (2008). “A Planning Graph Heuristic for Forward-Chaining Adversarial Planning”. In: *European Conference on Artificial Intelligence*. Vol. 8, pp. 921–922 (cit. on pp. 4, 25).

Blum, A. and M. L. Furst (1997). “Fast Planning Through Planning Graph Analysis”. In: *Artificial Intelligence* 90.1-2, pp. 281–300 (cit. on pp. 5, 40).

Bonet, B. and H. Geffner (2001). “Planning as Heuristic Search”. In: *Artificial Intelligence* 129, pp. 5–33 (cit. on p. 15).

Borrajo, D. (2013). “Multi-Agent Planning by Plan Reuse”. In: *Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 1141–1142 (cit. on p. 19).

Borrajo, D. and S. Fernández (2015). “MAPR and CMAP”. In: *Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*, pp. 1–3 (cit. on p. 19).

Boutilier, C. and R. Brafman (2001). “Partial-Order Planning With Concurrent Interacting Actions”. In: *Journal of Artificial Intelligence Research* 14.105, p. 136 (cit. on p. 21).

Bowling, M. H., R. M. Jensen, and M. M. Veloso (2003). “A Formalization of Equilibria for Multiagent Planning”. In: *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1460–1462 (cit. on pp. 28, 46).

Brafman, R. I., C. Domshlak, Y. Engel, and M. Tennenholtz (2009). “Planning Games”. In: *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 73–78 (cit. on pp. 3, 23).

Brandenburger, A. (2007). “Cooperative Game Theory”. In: *Teaching Materials at New York University* (cit. on pp. 21, 23).

Buzing, P., A. T. Mors, J. Valk, and C. Witteveen (2006). “Coordinating Self-Interested Planning Agents”. In: *Autonomous Agents and Multi-Agent Systems* 12.2, pp. 199–218 (cit. on p. 28).

Bylander, T. (1994). “The Computational Complexity of Propositional STRIPS Planning”. In: *Artificial Intelligence* 69.1, pp. 165–204 (cit. on pp. 13, 34, 115, 131).

Castillo, L., J. Fdez-Olivares, O. Garcia-Perez, and F. Palao (2005). “Temporal Enhancements of an HTN Planner”. In: *Conference of the Spanish Association for Artificial Intelligence*. Springer, pp. 429–438 (cit. on p. 17).

Cenamor, I., T. De La Rosa, and F. Fernández (2014). “IBACOP and IBACOP2 Planner”. In: *International Planning Competition 2014 Planner Abstracts*, pp. 35–38 (cit. on p. 16).

Chen, X. and X. Deng (2006). “Settling the Complexity of Two-Player Nash Equilibrium”. In: *47th Annual IEEE Symposium On Foundations of Computer Science*. IEEE, pp. 261–272 (cit. on pp. 114, 115).

Chen, Y., B. W. Wah, and C. Hsu (2006). “Temporal Planning Using Subgoal Partitioning and Resolution in SGPlan”. In: *Journal of Artificial Intelligence Research* 26, pp. 323–369 (cit. on p. 17).

Chevaleyre, Y., P. E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J. A. Rodríguez-Aguilar, and P. Sousa (2006). “Issues in Multiagent Resource Allocation”. In: *Informatica* 30.1, pp. 3–31 (cit. on p. 51).

Chien, S. and A. Sinclair (2011). “Convergence to Approximate Nash Equilibria in Congestion Games”. In: *Games and Economic Behavior* 71.2, pp. 315–327 (cit. on p. 131).

Clarke, E. H. (1971). “Multipart Pricing of Public Goods”. In: *Public Choice* 11.1, pp. 17–33 (cit. on pp. 27, 50).

Clement, B. (2005). *Multiagent Planning: A Survey of Research and Applications.* (Cit. on p. 1).

Coles, A., A. Coles, M. Fox, and D. Long (2010). “Forward-Chaining Partial-Order Planning”. In: *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 42–49 (cit. on p. 17).

Cox, J. and E. Durfee (2009). “Efficient and Distributable Methods for Solving the Multiagent Plan Coordination Problem”. In: *Multiagent Grid Systems* 5.4, pp. 373–408 (cit. on pp. 5, 21).

Cox, J. and E. Durfee (2004). “Efficient Mechanisms for Multiagent Plan Merging”. In: *Proceedings of the 3rd Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1342–1343 (cit. on p. 21).

Crosby, M., M. Rovatsos, and R. Petrick (2013). “Automated Agent Decomposition for Classical Planning”. In: *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 46–54 (cit. on p. 19).

Crosby, M. and M. Rovatsos (2011). “Heuristic Multiagent Planning With Self-Interested Agents”. In: *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Volume 1-3*, pp. 1213–1214 (cit. on p. 23).

De La Asunción, M., L. Castillo, J. Fdez-Olivares, Ó. García-Pérez, A. González, and F. Palao (2005). “SIADEX: An Interactive Knowledge-Based Planner for Decision Support in Forest Fire Fighting”. In: *AI Communications* 18.4, pp. 257–268 (cit. on p. 17).

De Weerdt, M., A. Bos, H. Tonino, and C. Witteveen (2003). “A Resource Logic for Multi-Agent Plan Merging”. In: *Annals of Mathematics and Artificial Intelligence* 37.1, pp. 93–130 (cit. on p. 21).

Decker, K. S. and V. R. Lesser (1992). “Generalizing the Partial Global Planning Algorithm”. In: *International Journal of Cooperative Information Systems* 01.02, pp. 319–346 (cit. on p. 20).

desJardins, M., E. Durfee, C. Ortiz, and M. Wolverton (1999). “A Survey of Research in Distributed Continual Planning”. In: *AI Magazine* 20.4, pp. 13–22 (cit. on p. 2).

desJardins, M. and M. Wolverton (1999). “Coordinating a Distributed Planning System”. In: *Artificial Intelligence* 20.4, pp. 45–53 (cit. on p. 21).

Domshlak, C., E. Karpas, and S. Markovitch (2010). “To Max or Not to Max: Online Learning for Speeding Up Optimal Planning”. In: *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. Atlanta, Georgia, pp. 1071–1076 (cit. on p. 16).

Dunne, P. E., S. Kraus, E. Manisterski, and M. Wooldridge (2010). “Solving Coalitional Resource Games”. In: *Artificial Intelligence* 174.1, pp. 20–50 (cit. on pp. 3, 23).

Durfee, E. H. (2001). “Distributed Problem Solving and Planning”. In: *Multi-Agent Systems and Applications*. Ed. by M. Luck, V. Mařík, O. Štěpánková, and R. Trappl. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 118–149 (cit. on pp. 2, 18).

Durfee, E. H. and V. R. Lesser (1991). “Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 21.5, pp. 1167–1183 (cit. on p. 20).

Edelkamp, S. (2003). “Taming Numbers and Durations in the Model Checking Integrated Planning System”. In: *Journal of Artificial Intelligence Research* 20, pp. 195–238 (cit. on p. 14).

Endriss, U., N. Maudet, F. Sadri, F. Toni, et al. (2006). “Negotiating Socially Optimal Allocations of Resources.” In: *Journal of Artificial Intelligence Research* 25, pp. 315–348 (cit. on p. 51).

Ephrati, E., M. Pollack, and S. Ur (1995). “Deriving Multi-Agent Coordination Through Filtering Strategies”. In: *Proceedings 14th International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann Publishers, pp. 679–687 (cit. on p. 21).

Erol, K., J. Hendler, and D. S. Nau (1994). “HTN Planning: Complexity and Expressivity”. In: *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*. Vol. 94, pp. 1123–1128 (cit. on p. 17).

Eyerich, P., R. Mattmüller, and G. Röger (2012). “Using the Context-Enhanced Additive Heuristic for Temporal and Numeric Planning”. In: *Towards Service Robots for Everyday Environments: Recent Advances in Designing Service Robots for Complex Tasks in Everyday Environments*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 49–64 (cit. on p. 17).

Fabrikant, A., C. Papadimitriou, and K. Talwar (2004). “The Complexity of Pure Nash Equilibria”. In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*. Chicago, IL, USA: ACM, pp. 604–612 (cit. on p. 130).

Fdez-Olivares, J., L. Castillo, J. A. Cózar, and O. García Pérez (2011). “Supporting Clinical Processes and Decisions by Hierarchical Planning and Scheduling”. In: *Computational Intelligence* 27.1, pp. 103–122 (cit. on p. 17).

Fikes, R. and N. Nilsson (1971). “STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving”. In: *Artificial Intelligence* 2.3, pp. 189–208 (cit. on pp. 14, 86).

Fišer, D., M. Štolba, and A. Komenda (2015). “MAPlan”. In: *Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*, pp. 8–10 (cit. on p. 19).

Fox, M. and D. Long (2003). “PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains”. In: *Journal of Artificial Intelligence Research* 20, pp. 61–124 (cit. on p. 14).

Friedman, J. W. and C. Mezzetti (2001). “Learning in Games by Random Sampling”. In: *Journal of Economic Theory* 98.1, pp. 55–84 (cit. on p. 120).

Galuszka, A. and A. Swierniak (2010). “Planning in Multi-Agent Environment Using Strips Representation and Non-Cooperative Equilibrium Strategy”. In: *Journal of Intelligent and Robotic Systems* 58.3, pp. 239–251 (cit. on p. 28).

Geffner, H. (2000). “Functional STRIPS: A More Flexible Language for Planning and Problem Solving”. In: *Logic-Based Artificial Intelligence, Jack Minker (Ed.), Kluwer* (cit. on p. 14).

Georgievski, I. and M. Aiello (2015). “HTN Planning: Overview, Comparison, and Beyond”. In: *Artificial Intelligence* 222, pp. 124–156 (cit. on p. 17).

Gerevini, A. and D. Long (2005). “Plan Constraints and Preferences in PDDL3”. In: *Technical Report, Department of Electronics for Automation, University of Brescia, Italy* (cit. on p. 14).

Gerevini, A. and I. Serina (2002). “LPG: A Planner Based on Local Search for Planning Graphs With Action Costs”. In: *Proceedings of the 6th International Conference on Artificial Intelligence Planning Systems*. AAAI Press, pp. 13–22 (cit. on pp. 16, 74, 76, 85).

Ghallab, M., A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins (1998). “PDDL - the Planning Domain Definition Language”. In: *AIPS-98 Planning Committee* (cit. on p. 14).

Ghallab, M., D. Nau, and P. Traverso (2004). *Automated Planning: Theory & Practice*. Elsevier (cit. on pp. 1, 13, 15, 99, 104).

Gillies, D. B. (1959). “Solutions to General Non-Zero-Sum Games”. In: *Contributions to the Theory of Games* 4.40, pp. 47–85 (cit. on pp. 22, 112).

Goemans, M., V. Mirrokni, and A. Vetta (2005). “Sink Equilibria and Convergence”. In: *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, pp. 142–154 (cit. on p. 129).

Grosz, B. J. and S. Kraus (1998). “The Evolution of SharedPlans”. In: *In A. Rao and M. Wooldrige, Foundations and Theories of Rational Agency*. Kluwer Academic Publishers, pp. 227–262 (cit. on p. 153).

Groves, T. (1973). “Incentives in Teams”. In: *Econometrica* 41.4, pp. 617–631 (cit. on pp. 27, 50).

Hadad, M., S. Kraus, I. B.-A. Hartman, and A. Rosenfeld (2013). “Group Planning With Time Constraints”. In: *Annals of Mathematics and Artificial Intelligence* 69.3, pp. 243–291 (cit. on p. 24).

Hart, S. and Y. Mansour (2010). “How Long to Equilibrium? the Communication Complexity of Uncoupled Equilibrium Procedures”. In: *Games and Economic Behavior* 69.1, pp. 107–126 (cit. on p. 130).

Helmert, M. (2003). “Complexity Results for Standard Benchmark Domains in Planning”. In: *Artificial Intelligence* 143.2, pp. 219–262 (cit. on pp. 115, 131, 136, 179).

Helmert, M. (2006). “The Fast Downward Planning System”. In: *Journal of Artificial Intelligence Research* 26.1, pp. 191–246 (cit. on pp. 15, 16, 136).

Helmert, M., P. Haslum, J. Hoffmann, and R. Nissim (2014). “Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces”. In: *Journal of the ACM* 61.3, 16:1–16:63 (cit. on p. 16).

Helmert, M., G. Röger, J. Seipp, E. Karpas, J. Hoffmann, E. Keyder, R. Nissim, S. Richter, and M. Westphal (2011). “Fast Downward Stone Soup”. In: *Proceedings of the 7th International Planning Competition (IPC-7)*, pp. 38–45 (cit. on p. 15).

Hoffmann, J. and B. Nebel (2001). “The FF Planning System: Fast Planning Generation Through Heuristic Search”. In: *Journal of Artificial Intelligence Research* 14, pp. 253–302 (cit. on p. 15).

Hrnčíř, J., M. Rovatsos, and M. Jakob (2015). “Ridesharing on Timetabled Transport Services: A Multiagent Planning Approach”. In: *Journal of Intelligent Transportation Systems* 19.1, pp. 89–105 (cit. on p. 28).

Jensen, R. M., M. M. Veloso, and M. H. Bowling (2001). “OBDD-Based Optimistic and Strong Cyclic Adversarial Planning”. In: *Proceedings of the 6th European Conference on Planning*, pp. 265–276 (cit. on p. 24).

Johnson, D. S., C. H. Papadimitriou, and M. Yannakakis (1988). “How Easy Is Local Search?” In: *Journal of Computer and System Sciences* 37.1, pp. 79–100 (cit. on p. 130).

Jonsson, A. and M. Rovatsos (2011). “Scaling Up Multiagent Planning: A Best-Response Approach”. In: *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS)* (cit. on pp. 5, 8, 10, 29, 30, 34, 95, 135, 136, 138, 141, 165, 178, 186).

Jordán, J. and E. Onaindía (2015). “Game-Theoretic Approach for Non-Cooperative Planning”. In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1357–1363 (cit. on pp. 30, 36, 66, 76, 95, 107).

Kameda, H., E. Altman, C. Touati, and A. Legrand (2012). “Nash Equilibrium Based Fairness”. English. In: *Mathematical Methods of Operations Research* 76.1, pp. 43–65 (cit. on p. 34).

Khouadjia, M. R., M. Schoenauer, V. Vidal, J. Dréo, and P. Savéant (2013). “Multi-Objective AI Planning: Comparing Aggregation and Pareto Approaches”. In: *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, pp. 202–213 (cit. on p. 17).

Knuth, D. E. and R. W. Moore (1975). “An Analysis of Alpha-Beta Pruning”. In: *Artificial Intelligence* 6.4, pp. 293–326 (cit. on p. 25).

Komenda, A., M. Stolba, and D. L. Kovacs (2016). “The International Competition of Distributed and Multiagent Planners (CoDMAP)”. In: *AI Magazine* 37.3, pp. 109–115 (cit. on pp. 10, 18, 135, 147).

Kovacs, D. L. (2011). *Complete BNF Description of PDDL3.1*. Tech. rep. (cit. on pp. 14, 18).

Larbi, R. B., S. Konieczny, and P. Marquis (2007). “Extending Classical Planning to the Multi-Agent Case: A Game-Theoretic Approach”. In: *Symbolic and Quantitative Approaches to Reasoning With Uncertainty, 9th European Conference, ECSQARU*, pp. 731–742 (cit. on pp. 5, 28, 34, 46).

Lesser, v., K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. Prasad, A. Raja, et al. (2004). “Evolution of the GPGP/-TAEMS Domain-Independent Coordination Framework”. In: *Autonomous Agents and Multi-Agent Systems* 9.1-2, pp. 87–143 (cit. on pp. 2, 20).

Luce, R. D. and H. Raiffa (1957). *Games and Decisions : Introduction and Critical Survey*. Wiley New York (cit. on p. 26).

Luis, N. and D. Borrajo (2014). “Plan Merging by Reuse for Multi-Agent Planning”. In: *2nd ICAPS Workshop on Distributed and Multi-Agent Planning (DMAP)*, pp. 38–44 (cit. on p. 19).

McKelvey, R. D., A. M. McLennan, and T. L. Turocy (2014). *Gambit: Software Tools for Game Theory, Version 13.1.2. http://www.gambit-project.org* (cit. on pp. 74, 87, 90).

Milchtaich, I. (1996). “Congestion Games With Player-Specific Payoff Functions”. In: *Games and Economic Behavior* 13.1, pp. 111–124 (cit. on p. 118).

Monderer, D. and L. S. Shapley (1996). “Potential Games”. In: *Games and Economic Behavior* 14.1, pp. 124–143 (cit. on pp. 116, 119, 120).

Mookherjee, D. and B. Sopher (1994). “Learning Behavior in an Experimental Matching Pennies Game”. In: *Games and Economic Behavior* 7.1, pp. 62–91 (cit. on p. 24).

Mors, A. T. and C. Witteveen (2005). “Coordinating Non Cooperative Planning Agents: Complexity Results”. In: *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pp. 407–413 (cit. on p. 28).

Muise, C., N. Lipovetzky, and M. Ramirez (2015). “Map-Lapkt: Omnipotent Multi-Agent Planning via Compilation to Classical Planning”. In: *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*, pp. 14–16 (cit. on p. 19).

Myerson, R. B. (1981). “Utilitarianism, Egalitarianism, and the Timing Effect in Social Choice Problems”. In: *Econometrica* 49.4, pp. 883–897 (cit. on p. 51).

Myerson, R. B. (2013). *Game Theory*. Harvard University Press (cit. on pp. 21, 25).

Nakhost, H. and M. Müller (2009). “Monte-Carlo Exploration for Deterministic Planning”. In: *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*. Vol. 9, pp. 1766–1771 (cit. on p. 16).

Nash, J. (1951). “Non-Cooperative Games”. In: *Annals of Mathematics* 54.2, pp. 286–295 (cit. on pp. 27, 34).

Nau, D., T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman (2003). “SHOP2: An HTN Planning System”. In: *Journal of Artificial Intelligence Research* 20, pp. 379–404 (cit. on p. 17).

Nau, D., Y. Cao, A. Lotem, and H. Munoz-Avila (1999). “SHOP: Simple Hierarchical Ordered Planner”. In: *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*. Stockholm, Sweden: Morgan Kaufmann Publishers Inc., pp. 968–973 (cit. on p. 17).

Nguyen, N. and R. Katarzyniak (2009). “Actions and Social Interactions in Multi-Agent Systems”. In: *Knowledge and Information Systems* 18.2, 133–136 (cit. on p. 1).

Nigro, N., D. Welch, and J. Peace (2015). *Strategic Planning to Implement Publicly Available EV Charching Stations: A Guide for Business and Policy Makers*. Tech. rep. Center for Climate and Energy Solutions (cit. on p. 152).

Nisan, N. and A. Ronen (2007). “Computationally Feasible VCG Mechanisms”. In: *Journal of Artificial Intelligence Research* 29.1, pp. 19–47 (cit. on pp. 96, 106).

Nisan, N., T. Roughgarden, E. Tardos, and V. v. Vazirani (2007). *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press (cit. on p. 125).

Nissim, R. and R. I. Brafman (2013). “Cost-Optimal Planning by Self-Interested Agents”. In: *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pp. 732–738 (cit. on pp. 27, 50).

Nissim, R., R. I. Brafman, and C. Domshlak (2010). “A General, Fully Distributed Multi-Agent Planning Algorithm”. In: *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1323–1330 (cit. on p. 29).

Ochs, J. (1995). “Games With Unique, Mixed Strategy Equilibria: An Experimental Study”. In: *Games and Economic Behavior* 10.1, pp. 202–217 (cit. on p. 24).

Osborne, M. J. and A. Rubinstein (1994). *A Course in Game Theory*. MIT Press (cit. on pp. 4, 21, 25, 45).

Papadimitriou, C. H. (1994). “On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence”. In: *Journal of Computer and System Sciences* 48.3, pp. 498–532 (cit. on p. 114).

Penberthy, J. and D. Weld (1992). “UCPOP: A Sound, Complete, Partial Order Planner for ADL”. In: *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR)*. Morgan Kaufmann Publishers, pp. 103–114 (cit. on pp. 16, 99).

Rawls, J. (1971). *A Theory of Justice*. Harvard Paperback. Harvard University Press (cit. on p. 51).

Richter, S. and M. Westphal (2010). “The LAMA Planner: Guiding Cost-Based Anytime Planning With Landmarks”. In: *Journal of Artificial Intelligence Research* 39.1, pp. 127–177 (cit. on pp. 16, 115, 136).

Rosenthal, R. W. (1973). “A Class of Games Possessing Pure-Strategy Nash Equilibria”. In: *International Journal of Game Theory* 2.1, pp. 65–67 (cit. on pp. 5, 29, 109, 116, 136).

Roughgarden, T. (2005). *Selfish Routing and the Price of Anarchy*. The MIT Press (cit. on p. 117).

Sailer, F., M. Buro, and M. Lanctot (2007). “Adversarial Planning Through Strategy Simulation”. In: *2007 IEEE Symposium on Computational Intelligence and Games*, pp. 80–87 (cit. on pp. 4, 25).

Sánchez-Garzón, I., J. Fdez-Olivares, E. Onaindía, G. Milla-Millán, J. Jordán, and P. Castejón (2013). “A Multi-Agent Planning Approach for the Generation of Personalized Treatment Plans of Comorbid Patients”. In: *Proceedings of the 14th Conference on Artificial Intelligence in Medicine*. Vol. 7885. Lecture Notes in Computer Science. Springer, pp. 23–27 (cit. on p. 17).

Sapena, Ó., E. Onaindia, and A. Torreño (2014). “FLAP: Applying Least-Commitment in Forward-Chaining Planning”. In: *AI Communications* 28.1, pp. 5–20 (cit. on p. 17).

Selten, R. (1975). “Reexamination of the Perfectness Concept for Equilibrium Points in Extensive Games”. In: *International Journal of Game Theory* 4.1, pp. 25–55 (cit. on pp. 23, 37, 50, 65).

Shehory, O. and S. Kraus (1998). “Methods for Task Allocation via Agent Coalition Formation”. In: *Artificial Intelligence* 101.1-2, pp. 165–200 (cit. on p. 23).

Shoham, Y. and K. Leyton-Brown (2009). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press (cit. on pp. 3, 4, 21, 22, 26, 45, 54, 60, 65, 112–114, 120, 125).

Štolba, M. and A. Komenda (2015). “MADLA: Planning With Distributed and Local Search”. In: *Competition of Distributed and Multi-Agent Planners (CoDMAP 2015)*, pp. 21–24 (cit. on p. 19).

Štolba, M., J. Tožička, and A. Komenda (2016). “Secure Multi-Agent Planning”. In: *Proceedings of the 1st International Workshop on AI for Privacy and Security*. PrAISe ’16. The Hague, Netherlands: ACM, pp. 1–8 (cit. on p. 18).

Stone, P., R. Brooks, E. Brynjolfsson, R. Calo, O. Etzioni, G. Hager, J. Hirschberg, S. Kalyanakrishnan, E. Kamar, S. Kraus, K. Leyton-Brown, D. Parkes, A. S. William Press, J. Shah, M. Tambe, and A. Teller (2016). *Artificial Intelligence and Life in 2030. One Hundred Year Study on Artificial Intelligence*. Tech. rep. Report of the 2015-2016 Study Panel. Stanford University, CA (cit. on p. 153).

Torralba, Á., C. L. López, and D. Borrajo (2016). “Abstraction Heuristics for Symbolic Bidirectional Search”. In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*. New York, New York, USA: AAAI Press, pp. 3272–3278 (cit. on p. 16).

Torreño, A., E. Onaindia, and Ó. Sapena (2014a). “A Flexible Coupling Approach to Multi-Agent Planning Under Incomplete Information”. In: *Knowledge and Information Systems* 38.1, pp. 141–178 (cit. on p. 154).

Torreño, A., E. Onaindia, and Ó. Sapena (2014b). “FMAP: Distributed Cooperative Multi-Agent Planning”. In: *Applied Intelligence* 41.2, pp. 606–626 (cit. on pp. 18, 137, 138).

Torreño, A., Ó. Sapena, and E. Onaindia (2015). “Global Heuristics for Distributed Cooperative Multi-Agent Planning”. In: *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI Press, pp. 225–233 (cit. on pp. 20, 124, 137, 138, 170).

Tožička, J., J. Jakubův, and A. Komenda (2015). “PSM-Based Planners Description for CoDMAP 2015 Competition”. In: *Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*, pp. 29–32 (cit. on p. 19).

Tožička, J., J. Jakubův, A. Komenda, and M. Pěchouček (2016). “Privacy-Concerned Multiagent Planning”. In: *Knowledge and Information Systems* 48.3, pp. 581–618 (cit. on p. 18).

Valenzano, R., H. Nakhost, M. Müller, J. Schaeffer, and N. Sturtevant (2012). “Arvandherd: Parallel Planning With a Portfolio”. In: *Proceedings of the 20th European Conference on Artificial Intelligence*. IOS Press, pp. 786–791 (cit. on p. 16).

Vallati, M., L. Chrpa, M. Grześ, T. L. McCluskey, M. Roberts, S. Sanner, et al. (2015). “The 2014 International Planning Competition: Progress and Trends”. In: *AI Magazine* 36.3, pp. 90–98 (cit. on p. 16).

Van Der Krogt, R. and M. De Weerdt (2005). “Self-Interested Planning Agents Using Plan Repair”. In: *ICAPS 2005 Workshop on Multiagent Planning and Scheduling*, pp. 36–44 (cit. on p. 27).

Vickrey, W. (1961). “Counterspeculation, Auctions, and Competitive Sealed Tenders”. In: *The Journal of Finance* 16.1, pp. 8–37 (cit. on pp. 27, 50).

Vidal, V. (2011). “YAHSP2: Keep It Simple, Stupid”. In: *Proceedings of the 7th International Planning Competition (IPC-7)* (cit. on p. 17).

Von Neumann, J. (1928). “Zur Theorie Der Gesellschaftsspiele”. In: *Mathematische Annalen* 100.1, pp. 295–320 (cit. on pp. 22, 24).

Von Neumann, J. and O. Morgenstern (2007). *Theory of Games and Economic Behavior*. Princeton University Press (cit. on pp. 3, 21, 22, 24).

Voorneveld, M., P. Borm, F. Van Megen, S. Tijs, and G. Facchini (1999). "Congestion Games and Potentials Reconsidered". In: *International Game Theory Review* 01.03n04, pp. 283–299 (cit. on p. 117).

Weerdt, M. D. and B. Clement (2009). "Introduction to Planning in Multiagent Systems". In: *Multiagent Grid Systems* 5.4, pp. 345–355 (cit. on p. 1).

Weld, D. (1994). "An Introduction to Least Commitment Planning". In: *AI Magazine* 15.4, p. 27 (cit. on p. 17).

Willmott, S., J. Richardson, A. Bundy, and J. Levine (1998). "An Adversarial Planning Approach to Go". In: *Proceedings of the International Conference on Computers and Games*. Ed. by H. J. Van Den Herik and H. Iida. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 93–112 (cit. on p. 25).

Willmott, S., J. Richardson, A. Bundy, and J. Levine (2001). "Applying Adversarial Planning Techniques to Go". In: *Theoretical Computer Science* 252.1, pp. 45–82 (cit. on p. 25).

Wilt, C. M. and W. Ruml (2011). "Cost-Based Heuristic Search Is Sensitive to the Ratio of Operator Costs". In: *Fourth Annual Symposium on Combinatorial Search*. AAAI Press (cit. on p. 116).

Wishart, J. (2013). *Utility Demand Charges and Electric Vehicle Supply Equipment* (cit. on p. 153).

Wooldridge, M., U. Endriss, S. Kraus, and J. Lang (2013). “Incentive Engineering for Boolean Games”. In: *Artificial Intelligence* 195, pp. 418–439 (cit. on pp. 96, 106).

Younes, H. and R. Simmons (2003). “VHPOP: Versatile Heuristic Partial Order Planner”. In: *Journal of Artificial Intelligence Research* 20, pp. 405–430 (cit. on p. 17).

Acronyms

AI Artificial Intelligence

AuPG Auction-Planning Game

BFS Breadth-First Search

BRP Best-Response Planning

BRPS Better-Response Planning Strategy

CGT Cooperative Game Theory

CoDMAP Competition of Distributed and Multi-Agent Planners

CoPG Coalition-Planning Game

DFS Depth-First Search

DTG Domain Transition Graph

EAV Electric Autonomous Vehicles

FD Fast Downward

FENOCOP Fair Equilibria in NOn-COoperative Planning

FF Fast Forward

FMAP Forward Multi-Agent Planning

GG General Game

HTN Hierarchical Task Network

IPC International Planning Competition

IPG Interaction Planning Game

MAP Multi-Agent Planning

MA-PDDL Multi-Agent Planning Domain Definition Language

MAS Multi-Agent System

MA-STRIPS Multi-Agent Stanford Research Institute Problem Solver

MH-FMAP Multi-Heuristic Forward Multi-Agent Planning

MSUS Multi-Symmetric Unsolvable Situation

NASA National Aeronautics and Space Administration

NCGT Non-Cooperative Game Theory

NE Nash Equilibrium

PDDL Planning Domain Definition Language

PLS Polynomial Local Search

PNE Pure strategy Nash Equilibrium

PO Pareto Optimal

POP Partial-Order Planning

PPAD Polynomial Parity Arguments on Directed graphs

PSPACE Polynomial Space

SG Scheduling Game

SPE Subgame Perfect Equilibrium

STRIPS Stanford Research Institute Problem Solver

VCG Vickrey–Clarke–Groves