Appl Intell (2018) 48:1020–1040
DOI 10.1007/s10489-017-1046-5

A better-response strategy for self-interested planning agents

Jaume Jordán[1] · Alejandro Torreño[1] · Mathijs de Weerdt[2] · Eva Onaindia[1]

Published online: 15 August 2017

Abstract When self-interested agents plan individually, interactions that prevent them from executing their actions as planned may arise. In these coordination problems, game-theoretic planning can be used to enhance the agents' strategic behavior considering the interactions as part of the agents' utility. In this work, we define a general-sum game in which interactions such as conflicts and congestions are reflected in the agents' utility. We propose a better-response planning strategy that guarantees convergence to an equilibrium joint plan by imposing a tax to agents involved in conflicts. We apply our approach to a real-world problem in which agents are Electric Autonomous Vehicles (EAVs). The EAVs intend to find a joint plan that ensures their individual goals are achievable in a transportation scenario where congestion and conflicting situations may arise. Although the task is computationally hard, as we theoretically prove, the experimental results show that our approach outperforms similar approaches in both performance and solution quality.

Keywords Planning · Game theory · Best-response · Better-response · Nash equilibrium

✉ Jaume Jordán
jjordan@dsic.upv.es

Alejandro Torreño
atorreno@dsic.upv.es

Mathijs de Weerdt
m.m.deweerdt@tudelft.nl

Eva Onaindia
onaindia@dsic.upv.es

[1] Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain

[2] EEMCS, Algorithmics, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands

1 Introduction

In some real-life planning problems, agents need to act strategically in order to achieve their goals. This is the case, for instance, of two agents that plan to simultaneously use a one-capacity resource, thus provoking a conflict between their plans at execution time. Instead, the construction of a coordinated plan would allow the agents to anticipate the conflict and build a joint plan with a better-utility outcome for both. In Multi-Agent Planning (MAP) with self-interested agents, decisions about what action to execute or when to execute an action are conditioned by possibly conflicting interests of the agents. We propose to address this problem with game-theory, the study of mathematical models of negotiation, conflict and cooperation between rational and self-interested agents [34]. Game-theoretic techniques are particularly suitable to the problem of designing a strategy (the agent's decision making model) that individual agents can use while negotiating – an agent will aim to use a strategy that maximizes its own individual welfare [17].

When agents that have their own incentives are involved in a MAP problem, there is a need for a stable solution plan, a plan from which none of the agents is willing to deviate during execution because otherwise it would only imply a loss of utility to some of them. In *coalitional planning*, self-interested agents create coalitions in order to share resources and cooperate on goal achievement because joining forces turns out to be more beneficial for reaching their goals [3, 8, 13]. Hence, in cooperative game-theoretic models such as coalitional planning, self-interested agents build their plans on the basis of a cooperative behavior and exploitation of synergies with the other agents. This breaches the principle of independence if the agents wish to maintain their autonomy. In contrast, when agents plan autonomously in a strictly-competitive setting, the problem is known as *adversarial planning*, and it is commonly modeled as a zero-sum game [34]. In these problems, agents try to prevent others from reaching their goals since each agent's gain of utility is exactly balanced by the losses of the other agents [2, 7].

However, some MAP problems do not fit as coalitional or adversarial planning. Between these two game-theoretic planning settings, there is a large number of problems in which self-interested agents work independently on their planning problem (no coalition formation) and the joint execution of their plans is not regarded as a strict competition. In non-strictly competitive settings, agents have conflicting but also complementary interests and they are willing to cooperate with the aim of finding a solution that benefits all of them. The middle ground between coalitional and adversarial planning is a largely unstudied problem, which we will refer to as *non-cooperative MAP*. This type of problems are modeled as non-zero-sum or general-sum games, where the winnings and losses of all agents do not add up to zero and win-win situations can be reached by seeking a common solution that accommodates the local solutions of all agents. In other words, although agents are self-interested and non-cooperative, they also wish to achieve a stable (equilibrium) joint plan that ensures their plans are executable (by avoiding any conflict). Some real-life problems that involve agents sharing resources to accomplish their plans or the coordination of traffic flow to avoid collisions embody non-cooperative MAP scenarios.

Finding stable multi-agent plans can be done with the Best-Response Planning (BRP) proposed in [19]. This approach solves *congestion planning games* through an iterative plan-improvement process that initiates with an executable initial joint plan. Since the initial joint plan may use synergies between agents' plans, agents will be compelled to stick to some actions, which may be against their strategic behavior and private interests. Additionally, due to this agent dependency, convergence to an equilibrium is not guaranteed. Other techniques such as plan merging would solve the problem of conflict interactions [35], but, likewise, making use of synergies is not compliant with self-interested agents that plan autonomously in scenarios with competitive relationships. The theoretical approach in [20] 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. In [20], the strategies of the agents are limited to a given subset of plans, congestion situations are not considered and the complexity of the task renders the calculation of all the equilibria intractable. All in all, there does not exist computational proposals in the non-cooperative MAP literature that synthesize a joint plan while strategically resolving conflicts and congestion interactions among the agents' plans.

In this work, we present the Better-Response Planning Strategy (BRPS), a game-theoretic *non-cooperative MAP* approach that finds a joint plan for a set of self-interested agents in problems with congestion and conflicting situations. In BRPS, agents adapt their plans to the other agents' plans in an iterative cost-minimization process in which the interactions among the agents imply a cost rise that is modeled in a single individual agent's cost function. We consider both congestion and conflicts as part of the agents' cost functions. This way, agents are incentivized to avoid conflicts by applying the so-called *taxation schemes* [25, 36], in which a third party taxes agents incurring conflicts to guarantee the feasible execution of the agents' plans.

BRPS is a general-purpose non-cooperative MAP approach capable of solving different planning problems. Particularly, we designed an Electric Autonomous Vehicles (EAV) [24] domain to perform a comprehensive experimental evaluation of our approach. In this domain, agents are electric self-driving taxi companies in a smart city. Since EAVs are regarded as rational and self-interested entities, conflicting plans including vehicles attempting to reach a charger at the same time can be avoided by coordinating the actions of their plans. Thus, agents can opt for visiting other locations before the charger or waiting until the charger is available, depending on the impact of each strategy in their utility values. Ultimately, the objective is to find a solution that accommodates all the local solutions and allows agents to achieve their goals with the maximum utility possible.

This work is organized as follows. Next section presents the planning problem in which all elements that affect the agents' utility are defined. Section 3 formalizes the planning problem as a game-theoretic approach, the *Interaction Planning Game* (IPG), and we show the complexity of the task as well as under which conditions the IPG is a potential game. In Section 4, we present BRPS, the better-response planning strategy to solve the IPG, and we analyze the convergence to equilibrium solutions. Section 5 introduces the EAVs domain which features both conflicts and congestion. In Section 6, we show an application example of BRPS in the EAVs domain and different experimental results, including a comparative with the BRP approach of [19]. The last section presents the conclusions.

2 Planning framework

A MAP task consists of a set, $\mathcal{AG}$, of n rational self-interested planning agents where each agent i synthesizes a plan π^i to accomplish its individual planning task. The utility that π^i reports to agent i may be jeopardized at execution time due to the interactions with the actions of the other plans. Thus, agents are willing to reach an equilibrium that guarantees their best possible solution jointly with the others'.

For the sake of clarity, we briefly name all of the agents costs that will be used 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.

A MAP task is modeled as deterministic planning problem in a fully-observable environment. The world consists of a set of state variables ($\mathcal{V}$) each associated to a finite domain ($\mathcal{D}_v$, $v \in \mathcal{V}$). A variable takes a value of its domain through a variable assignment ($v := d, d \in \mathcal{D}_v$). A state S is a total variable assignment over $\mathcal{V}$. Each agent has its own view of the world which may be totally or partially shared with the other agents.

Definition 1 The ***planning task of an agent*** $i \in \mathcal{AG}$ is a tuple $\mathcal{T}^i = \langle \mathcal{V}^i, \mathcal{I}^i, \mathcal{A}^i, \mathcal{G}^i \rangle$, where $\mathcal{V}^i$ is the set of state variables known to agent i; $\mathcal{I}^i$ is the initial state over $\mathcal{V}^i$; $\mathcal{A}^i$ is a finite set of actions over $\mathcal{V}^i$ describing the performable operations by agent i; and $\mathcal{G}^i$ a formula over $\mathcal{V}^i$ describing the goals of the agent.

A *planning action* of $\mathcal{A}^i$ is a tuple $\alpha^i = \langle pre(\alpha^i), eff(\alpha^i), cost(\alpha^i) \rangle$, 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 $cost(\alpha^i)$ is a numeric value that denotes the cost of executing α^i. 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.

Given two agents i and j, G^i and G^j will not contain antagonist goals since it would be otherwise an adversarial planning task. On the other hand, G^i and G^j are generally disjoint sets ($\mathcal{G}^i \cap \mathcal{G}^j = \emptyset$) because the goal formula ($v = d$) are defined over different sets of variables, $\mathcal{V}^i$ and $\mathcal{V}^j$, respectively. Even though agents could share one same variable, this is not typically the case for agents that solve independent planning tasks. Agents are assumed to solve their assigned goals individually without any assistance or synergy.

Agents develop solutions for their own tasks in the form of *partial-order plans*.

Definition 2 A ***partial-order plan*** of an agent $i \in \mathcal{AG}$ is a pair $\pi^i = \langle \mathcal{X}^i, \prec \rangle$, where $\mathcal{X}^i \subseteq \mathcal{A}^i$ is a nonempty subset of the actions of agent i and $\prec$ is a strict partial order on $\mathcal{X}^i$.

Every strict partial order is a directed acyclic graph. Two unordered actions α^i_j and α^i_k of a plan π^i are executable 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. The set of topological sorts of π^i 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 utility of π^i is measured as the utility that achieving $\mathcal{G}^i$ reports to i. Since two different plans that achieve $\mathcal{G}^i$ will bring i the same utility, agents will use the cost of executing a plan, denoted as $costP(\pi^i)$, to differentiate plans. The term $costP(\pi^i)$ measures two aspects:

– Cost of the actions in π^i. $cost(\alpha^i), \forall \alpha^i \in \pi^i$ denotes 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.
– Finish time of π^i. For some agents, achieving the goals sooner or later will have a different impact in the agent's utility. If two plans have the same action cost, agents will most likely prefer the one that finishes earlier.

The particular evaluation of the action cost and finish time of π^i will depend on the context, infrastructure and individuality of i. $costP(\pi^i)$ weights all the relevant parameters to agent i, representing how costly is for i to execute π^i.

Definition 3 A ***joint plan*** is a tuple $\Pi = \langle \pi^1, \pi^2, \ldots \pi^n, \prec_{\mathcal{AG}} \rangle$ where $\prec_{\mathcal{AG}}$ is a set of inter-agent orderings over the actions of the partial-order plans of the n agents.

We use the notation $\Pi^{-i} = \langle \pi^1, \ldots, \pi^{i-1}, \pi^{i+1}, \ldots, \pi^n \rangle$ 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.

2.1 Cost of integrating a plan in a joint plan

Ideally, executing π^i along with Π^{-i} would only charge $costP(\pi^i)$ to agent i. However, integrating π^i in Π^{-i} may cause *interactions* (conflicts or congestions) between the actions in π^i and the actions in Π^{-i} and solving these interactions make agents incur an additional cost. The purpose of agent i is to examine how costly it is to integrate π^i in Π^{-i} is.

Conflict Interactions A *conflict* interaction is a situation between the plans of two agents in which executing an action of one agent in some specific order may prevent the other one from executing one of its actions.

In a partial-order plan, a particular type of precedence relation $\alpha \prec \beta$ exists if a supporting effect of α ($v := d \in eff(\alpha)$) is used to satisfy a precondition of β ($v = d \in pre(\beta)$). We will denote such a causal relationship as $\alpha \prec_{\langle v,d \rangle} \beta$.

Definition 4 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 4 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 partial-order planning 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 [11]. 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 can adopt the role of conflict solvers. A conflict $c = \langle \gamma^i, \alpha^j, \beta^j \rangle$ is a **solvable conflict** by agent i (resp. j) by setting $\beta^j \prec_{\mathcal{AG}} \gamma^i$ (resp. $\gamma^i \prec_{\mathcal{AG}} \alpha^j$) as long as the newly introduced precedence relation is consistent with the sets $\prec$ and $\prec_{\mathcal{AG}}$ of π^j (resp. π^i). Note that an agent is only allowed to insert orderings that keep the plan of the other agent unaltered. Agents seek their own benefit but not at the cost of provoking conflicts to others because this would have a negative impact in all the involved agents.

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 ordering $\prec_{\mathcal{AG}}$ is set to an action γ^i of π^i, the time step of γ^i and its successors must be now re-calculated 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 will cause an impact in the integration cost of agent i. The delay cost caused by solving the inter-agent conflicts is included in $costS(\pi^i, \Pi^{-i})$.

Our approach also accounts for unsolvable conflicts and charges the agent accordingly in order to encourage the agent to deviate from such a conflicting situation and to 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 $penalty^i$.
- *An unsolvable outgoing conflict* $\langle \gamma^i, \alpha^j, \beta^j \rangle$ of agent i affects the feasibility of π^j. In a general-sum and non-strictly competitive game, an agent is taxed if its plan provokes an unsolvable conflict. We use a taxation scheme [25, 36] that imposes tax^i to agent i for obstructing the execution of the plan of another agent j.

The cost of a joint plan with unsolvable conflicts must surpass the cost of a plan with no conflicts or with solvable conflicts because it is the worst outcome for any agent. Consequently, 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. Both $penalty^i$ and tax^i are set to a value cc^i that exceeds the cost of the worst possible conflict-free joint plan. 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, agent i will be charged with $cc^i \cdot |U|$, where $|U|$ is the number of unsolvable conflicts. We will denote such a cost by $costU(\pi^i, \Pi^{-i})$.

Congestion interactions 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 [29]. 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 formulae $v = d, v \in \mathcal{R}$ in their preconditions. Moreover, the cost of a congestion may differ across the agents involved in it since each agent has its individual cost function, which makes our approach more realistic. Given $\mathcal{R} = \{r_1, \ldots, r_m\}$, 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 in a joint plan. $\mathcal{N} : \mathcal{J} \times \mathbb{N} \times \mathcal{R} \rightarrow \mathbb{N}$ returns the number of actions that use resource r at time t in a given joint plan $\Pi \in \mathcal{J}$ (where $\mathcal{J}$ is the set of all possible joint plans). Therefore, the congestion cost incurred by agent i is defined as $costG(\pi^i, \Pi^{-i}) = \sum_{t=0}^{finish(\Pi)} \sum_{r \in \mathcal{R}} \mathcal{C}_r^i(\mathcal{N}(\Pi, t, r))$.

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. 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. Therefore, we define as $costG(\pi^i, \Pi^{-i})$ the cost of the non-resolved congestion interactions of π^i.

Finally, the total cost of integrating π^i into Π^{-i} is:

$$\begin{aligned} costTotal(\pi^i, \Pi^{-i}) = {} & costP(\pi^i) + costG(\pi^i, \Pi^{-i}) \\ & + costS(\pi^i, \Pi^{-i}) \\ & + costU(\pi^i, \Pi^{-i}) \end{aligned} \tag{1}$$

The net utility that a plan π^i reports to agent i will be the utility of achieving $\mathcal{G}^i$ minus $costTotal(\pi^i, \Pi^{-i})$.

Note that $costTotal(\pi^i)$ computes the cost of realization of π^i along with the plans of the other agents and this is the only factor that drives the agents' decision-making since all individuals in a game-theoretic setting are aimed to a strategy that maximizes their own individual welfare. Social cost factors such as trust and reputation are used to assess the cost of decisions other than purely economic impact. Particularly, how trustworthy an agent is when executing a plan could have an impact in the cost assessment of the agents. However, this is not applicable in this context because, as we will see in Section 4, the formal game-theoretic properties guarantee that no agent will deviate from a stable joint solution plan. Social cost factors are applicable in other type of negotiation frameworks such as argumentation-based approaches.

Definition 5 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 $\bigcup_{i\in\mathcal{AG}} \mathcal{G}^i$.

3 Interaction planning game

An Interaction Planning Game (IPG) is a general-sum game to solve the problem of multiple self-interested agents all wanting to execute their plan in the same environment. In a general-sum game, agents' aggregate gains and losses can be less or greater than zero, meaning that agents do not try to minimize the others' utilities. In the IPG, agents are self-interested but not strictly competitive so the aim of an agent is to seek a plan which does not provoke a conflict with the other agents' plans. Otherwise, this 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.

An agent i solves its task $\mathcal{T}^i$ by generating a plan π^i with actions from $\mathcal{A}^i$ that achieves its goals in $\mathcal{G}^i$.

Definition 6 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 $u^i : \pi^i, \Pi \rightarrow \mathbb{R}$ 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 $\Pi = \langle \pi^1, \ldots, \pi^{i-1}, \pi^i, \pi^{i+1}, \ldots, \pi^n \rangle$.

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 Nash Equilibrium (NE) (see [30, Chapter 3] for more information).

Definition 7 An ***IPG solution*** is a conflict-free solution joint plan Π^* (as defined in Def. 5) which is a NE of the IPG.

The complexity of finding a NE in the IPG is PPAD-hard (Polynomial Parity Arguments on Directed graphs) [27] since computing a NE, even in a 2-player game, is PPAD-complete [5] 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 [30, Chapter 4].

Theorem 1 *Computing a NE for an IPG is PPAD-hard even for single-action plans.*

Proof The idea of this proof is to use a reduction from general-sum finite games. For this class of games, any strategy of a player/agent i can be translated in polynomial time to a task $\mathcal{T}^i$ of the IPG. This is done by mapping the strategies of any general-sum game to single-action plans of the IPG. 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. □

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. In the general case, planning in propositional STRIPS is PSPACE-complete [4] and cost-optimal planning has proven even more difficult to solve in practice [1].

Theorem 2 *IPG is PSPACE-hard even with just one agent.*

Proof The sketch of this proof is to 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 an IPG with task $\mathcal{T}^i$ and $\mathcal{AG} = \{i\}$. Then, solving this IPG is only about computing single-agent plans that solve $\mathcal{T}^i$. □

Monderer and Shapley [23] found a more general class than congestion games 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). 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:

$$\Phi(\Pi) = \sum_{i \in \mathcal{AG}} costTotal(\pi^i, \Pi) \tag{2}$$

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 individual payoff functions, a game is not potential anymore as it is proven in [22]. 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 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 case (a) and (b). □

For potential games, convergence to PNE by best/better response is guaranteed [23]. 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 4.3. In Section 6, we describe experimental results that aim to evaluate convergence properties by better-response dynamics in a concrete domain that do not meet the conditions from the above Theorem 3. 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 cost functions of the agents, which makes our model more complete.

4 Better-response planning strategy

In this section, we explain the **Better-Response Planning Strategy** (BRPS) applied to the IPG, the search process of BRPS, the convergence of BRPS to a Pure strategy Nash Equilibrium (PNE), and we present a discussion about the complexity of the BRPS in the IPG.

4.1 BRPS process

Better-response dynamics draw upon the properties defined for best-response dynamics. Particularly, we know that any finite potential game [23] 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* [30, Chapter 6]. Additionally, a better-response strategy can be implemented by an agent by randomly sampling another plan until one is found with less cost than the current plan's, and this does not require the agent to know the cost of every plan in its search space [10]. In our planning context, we use better response instead of best response since agents do not need to find the best plan at each iteration, which may be computationally intractable.

Our BRPS is a process in which each agent i iteratively revises its plan π_x^i in the joint plan Π, and switches to another plan π_y^i which integrated in Π^{-i} reports i a utility better than π_x^i. Before starting the process, an empty joint plan $\Pi = \varnothing$ and an arbitrary order between the agents in $\mathcal{AG}$ are established. 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, BRPS has reached a convergence point in which the current joint plan is a PNE.

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 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 1 show the utilities of the 16 joints plans that result from combining the four plans of each agent. The terms cc^1 and cc^2 denote the cost of the penalty/tax charged to agent 1 and 2, respectively, for the unsolvable conflicts in the joint plans. Table 1 shows 7 solution joint plans, four of which displayed in bold are PNE. If BRPS obtains the joint plan $\langle\pi_4^1, \pi_4^2\rangle$ with utilities (−4,−4) at some point of the process, we can say BRPS has reached convergence because no agent is able to come up with a better plan without conflicts given the plan of the others and so the utility of none of the agents can be improved. The joint plan $\langle\pi_4^1, \pi_4^2\rangle$ is PNE but it is not Pareto Optimal (PO) whereas the rest of PNE plans are all PO. 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, 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 (1) 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 (1) and if the cost is greater or equal than $upper^i$ then the

Table 1 Two agents with conflicts example

	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**

PNE in bold

node is pruned. Otherwise, when the node already holds all of the actions of the plan π^i_y and if $costTotal(\pi^i_y, \Pi^{-i}) < upper^i$, then the search stops because a better response has been found. In this case, $\Pi' = \langle \pi^1, \ldots, \pi^i_y, \ldots, \pi^n \rangle$ 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^i_x 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.

4.2 Search procedure

In BRPS, each agent i implements an individual A* search procedure that progressively generates better responses; i.e., individual plans that solve its task $\mathcal{T}^i$, and integrates them into the current joint plan. In one BRPS iteration, agent i calculates $upper^i = costTotal(\pi^i_x, \Pi^{-i})$ as the cost of its current proposal in the joint plan, removes π^i_x, and autonomously launches an A* search to find and integrate a better response, π^i_y, 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: $\pi^i_{y_0} = \langle \mathcal{X}^i = \emptyset, \prec \rangle$. We will denote such a combination as $\Pi^{-i} \circ \pi^i_{y_0}$.

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 $\Pi^{-i} \circ \pi^i_{y_0}$, its successor nodes will contain $\Pi^{-i} \circ \pi^i_{y_1}$, where $\pi^i_{y_1} = \langle \mathcal{X} = \{\alpha^i_1\}, \prec \rangle$; a successor of $\Pi^{-i} \circ \pi^i_{y_1}$ will be $\Pi^{-i} \circ \pi^i_{y_2}$, where $\pi^i_{y_2} = \langle \mathcal{X} = \{\alpha^i_1, \alpha^i_2\}, \prec \rangle$; and so on until a node which contains $\Pi^{-i} \circ \pi^i_y$ is found. In other words, each node of the tree successively adds and consistently supports a newly added action until a node that contains a complete plan π^i_y 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 2, these elements are merely used for conflict resolution.

The search is aimed at finding a plan for agent i without conflicts with the other agents' plans. 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 [33] 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.

4.3 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 lead to this situation and we show that in the absence of this type of conflicts, 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 [30, Chapter 6], [26, Chapter 19].

Corollary 1 *Better-response dynamics of an IPG always converges to a PNE if the potential game property holds.*

As we explained in Theorem 3, the potential game property with the potential function Φ only holds under some assumptions. However, even without these assumptions, and considering the cost functions of the agents as defined in (1) ($costTotal$, where the agents consider their own plans, congestions, unsolved conflicts, and delays of solvable conflicts and/or congestions), the IPG with better-response dynamics will converge to a PNE in most cases.

4.3.1 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 happens in 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 improves u^i or u^j due to the existence of conflicts.

Definition 8 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

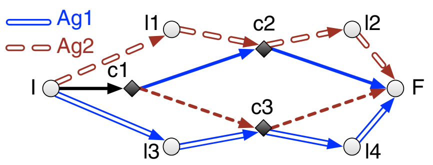

Fig. 1 Multi-symmetric unsolvable situation example

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 1 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 Fig. 1. 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 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 2 Multi-symmetric unsolvable situation

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

PNE in bold

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 Fig. 1, 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 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 1, the IPG is a potential game (under some assumptions) with an associated ordinal potential function Φ of (2) 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.

4.3.2 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 agent's plan's 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 potential game property is broken. When the IPG is no longer 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 Section 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* [12].

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 [12]. Therefore, we announce the following proposition:

Proposition 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 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. □

Although 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 Section 6.

4.4 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 [18]. 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 [9]. Moreover, the lower bound on the speed of convergence to NE is exponential in the number of players [14]. 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 1.

While these are good news for the IPG in general, we note that computing a strategy for an agent implies planning, which is PSPACE-complete in the general case [4], as we pointed out in Theorem 2. However, planning complexity can be lower for some planning domains as it is shown by [15]. 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-10STRIPS`, and `MICONIC-10-SIMPLE`). In contrast, optimal planning is always NP-complete. This is one of the reasons why our 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 our 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 [6] must be guaranteed in order to ensure PLS-completeness of the IPG with the BRPS approach.

Proposition 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-10-STRIPS`, or `MICONIC-10-SIMPLE`, for which a non-optimal plan can be computed in polynomial time, as specified in [15]. 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 good 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 Case study: electric autonomous taxis in a Smart city

In this section, we present a case study in which various autonomous taxi companies (agents) seek their own benefit without necessarily jeopardizing the plans of other taxi companies in the context of a clean, coordinated and harmonic smart city. We designed an Electric Autonomous Vehicle (EAV) domain with two main purposes in mind: a) dealing with a challenging problem in the near future and b) testing a planning domain for self-interested agents which consider both congestion and conflicts.

The EAV domain resembles the popular game 'Battle of the Sexes', where a player receives a reward which depends on how much preferable one activity is to the player plus an additional reward if the other player also chooses the same activity; i.e., if the activities of both players are coordinated. However, coordinating the interests of autonomous agents (plans of electric self-driving taxi companies) in a collective environment (the city) brings about situations of congestion and negative interactions between the actions of the agents (e.g., conflicts for the usage of a particular resource) which may render the plans unfeasible.

In order to properly motivate our EAV case study, Fig. 2 shows the area covered by a number of taxi companies in a European city. The route of a `taxi` is determined by the streets (black edges) it traverses. A street is defined by the two `junctions` (gray circular nodes) it connects. Across the city, there are several `chargers` (green squares) in which 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` on the network map 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 [31] that incorporates explicit support of congestion interactions. Agents or taxi `companies` 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` in `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`.

A `charger` is accessible by a single `taxi` at a time. Since `taxis` act in the same environment, a ***charging station occupancy conflict*** occurs when a `taxi` comes across an occupied `charger`. In this case, 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. delaying the `charge` action to avoid the conflict).

Congested traffic flow directly affects the cost of the `taxis`' actions. We identify two different types of congestion:

– ***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 congestion. Consequently, the cost associated to the `drive` action of each `taxi` will increase. Agents should then consider traffic congestion when selecting the routes of their `taxis`.
– ***Electricity network congestion***. When `taxis` intend to recharge their batteries simultaneously at different chargers of the same electricity network, prices will raise due to a peak demand, also leading to an electricity shortage. Thus, `company` agents will be penalized if they get involved in an electricity network congestion.

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,

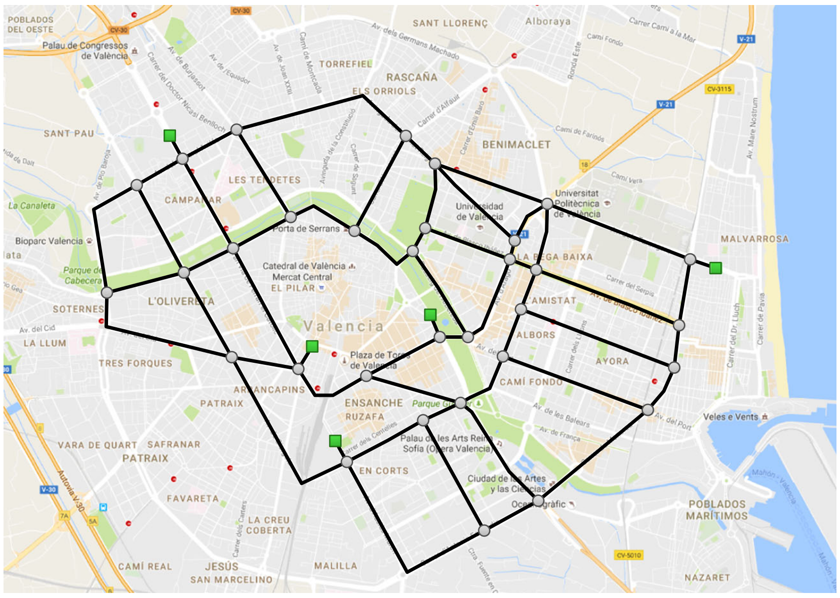

Fig. 2 Smart city map example

a conflict makes the involved plans 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.

5.1 BRPS problem example

In order to illustrate the behavior of BRPS when solving a planning problem with self-interested agents, Fig. 3 shows a simple example based on the EAV domain. This example features three taxi companies, `Company1`, `Company2`, and `Company3`, each of them having a single vehicle (`t1`, `t2` and `t3`) and one passenger to transport (`p1`, `p2` and `p3`). There are four connected junctions `j1` to `j4` and two chargers `c1` and `c2` in the same electricity network `n1` which are accessible from `j1` and `j2`, respectively (see Fig. 3). Taxis `t1` and `t3` start at junction `j1`, and `t2` starts at `j2`. The batteries of the taxis are initially empty (`level l0`), and their maximal capacity is `l2`.

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

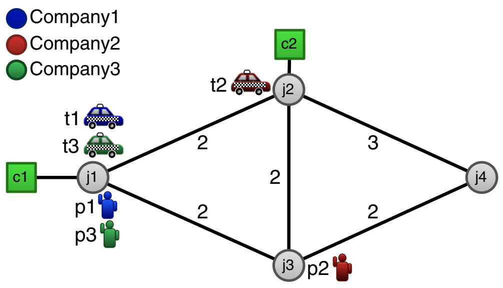

Fig. 3 Problem example representation

`drive` actions, whose cost depends on the length of the street, as shown in the edges of Fig. 3. 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 $\mathcal{N}(\Pi, t, r)$, 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 3 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 transport `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, with the final plans integrated in the solution joint plan.

As explained in Section 4, 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.

Table 3 Individual agents' plans

t	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
Costs	$costP(\pi_1^1) = 8$	$costP(\pi_1^2) = 8$	$costP(\pi_1^3) = 8$

- **Iteration 1**:
  - Company1 generates its plan π_1^1 with $costTotal(\pi_1^1, \Pi^{-1}) = 8$ (see Table 3). 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. An electricity network congestion is present 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; i.e., $costTotal(\pi_1^1, \Pi^{-1}) = 8 + 2 + 2 = 12$. 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 unable 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 4 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 Experimental results

In this section, we test the performance of BRPS through a set of problem instances of the EAV domain presented in Section 5. We provide some details of the BRPS implementation, including a brief discussion about the underlying MAP technologies it uses, in Section 6.1. Next, Section 6.2 analyzes the experimental results, comparing the performance of BRPS against the state-of-the-art BRP approach [19]. In Section 6.3, we analyze the strategic behavior adopted by the agents with the better-response dynamics of BRPS. Finally, Section 6.4 provides a comprehensive analysis of the results obtained by each BRPS agent.

6.1 BRPS implementation details

BRPS is implemented on top of a modified version of the MH-FMAP satisficing planner [33]. BRPS draws upon the features of MH-FMAP, including its multi-agent data structures, the communication infrastructure and message passing protocols among agents, the privacy model [32], and the heuristic functions [33]. As it was mentioned in Section 5, we designed an extension to the PDDL-based[1] MAP language presented in [31] to include explicit support of congestion interactions.

[1]PDDL: Planning Domain Description Language [21]

An agent i of BRPS uses MH-FMAP to individually synthesize the plan (response) that will be integrated in the current joint plan Π^{-i}. The search of an agent is efficiently guided by a) the cost of its previous response, which is used as a threshold to prune parts of the tree; and b) the heuristics of MH-FMAP, which have been adapted to deal with the cost functions of the agents. Moreover, the BRPS search of an agent can return a plan with unsolved conflicts.

6.2 Comparative evaluation of BRPS and BRP in the EAV domain

In non-cooperative MAP, particularly in game-theoretic planning, there are hardly available domain-independent frameworks. One notable exception is the Best-Response Planning (BRP)[2] approach presented in [19]. BRP is a game-theoretic planning approach with the following characteristics:

1. It is specifically designed to compute equilibria in *congestion games*.
2. It requires an initial conflict-free joint plan which is computed offline by a *cooperative MAP solver*; i.e., the plan is calculated regardless the private incentives of the agents and synergies among the agents' plans may appear.[3] The joint plan comprises one plan per agent that achieves the corresponding goals.
3. It 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.
4. It applies best response instead of better response, which entails a more costly plan generation for the agents.
5. It 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.

These features reveal that BRP and BRPS show a similar behavior and so they are comparable. We created a synthetic benchmark of the EAV domain that includes 25 multi-agent problems of growing complexity. Table 5 shows the problem setup of this benchmark. The columns of Table 5 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 5, the number of company agents per problem ranges between 2 and 6: the first 5 problems, p1-2 to p5-2, include two agents; the next 5 problems, from p6-3 to p10-3, feature 3 agents, and so on. In each 5-problem

[2]We used optimal Fast-Downward [16] as the underlying individual planner for BRP agents, since it was the best-performing setting in our tests.

[3]Due to implementation limitations of BRP, it is not possible to use a cooperative MAP solver as stated in [19]. Therefore, we used the satisficing LAMA planner [28], which performed better than Fast-Downward to compute the initial conflict-free joint plan.

Table 4 Resulting IPG solution joint plan Π

t	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
Costs	$costTotal(\pi_2^1, \Pi^{-1}) = 9 + 2 = 11$	$costTotal(\pi_1^2, \Pi^{-2}) = 8 + 2 = 10$	$costTotal(\pi_1^3, \Pi^{-3}) = 8 + 2 * 5 = 18$

block, the 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.

Table 5 Problem setup of the benchmark of test for the electric autonomous taxis domain

	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

The experimental results for both approaches are summarized in Table 6.[4] The first three columns of each planner refer to the number of actions, 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 Section 5.1. 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 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 to an IPG solution requires significantly larger computation times in these problems.

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 do not reflect the time needed for the calculation

[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 Experimental results for different problem setups of the EAV domain with random order between agents

	BRP					BRPS				
	Acts	Fh	Cost	Iters	Time	Acts	Fh	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

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 4.3.

BRPS proves to be particularly efficient at optimizing the finish time of the solution joint plans. Even though many of the solution plans contain a large number of actions, the finish time 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 finish time and cost. This is also supported by the partial-order reasoning mechanism of the planner MH-FMAP [33]. 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 finish time, which in turn penalizes the cost of the solution plans.

We can also observe in Table 6 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. **Finish time**: The underlying MAP machinery of BRPS efficiently enforces parallelism among the agents' actions, keeping the finish time 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.

6.3 Analyzing 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 5.1 and depicted in Fig. 3 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 7 show the number of actions, finish time, and cost of the plans of each `company` agent in the six different settings.

In the following, we analyze the six configurations used in this experiment and the results summarized in Table 7:

- **Setting 1**: This is the **original setting** of the problem as presented in Section 5.1, where the **cost of a delay of one time step is 5**. The solution plan for this setting is shown in Table 4. As explained in Section 5.1, 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

Table 7 Strategic behavior analysis for different cost functions

	`Company1`			`Company2`			`Company3`		
Setting	Act	Fh	Cost	Act	Fh	Cost	Act	Fh	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

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 8. 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 finish time 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).

– **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.

6.4 Influence of the order 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 4 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, finish time and cost of a specific agent for all problems of Table 6 with respect to the best individual plan of the agent, which is computed exhausting the search space of the agent.

Table 8 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
Costs	$costTotal(\pi_1^1, \Pi^{-1}) = 8$	$costTotal(\pi_3^2, \Pi^{-2}) = 10 + 1 * 5 = 15$	$costTotal(\pi_1^3, \Pi^{-3}) = 8 + 4 * 5 = 28$

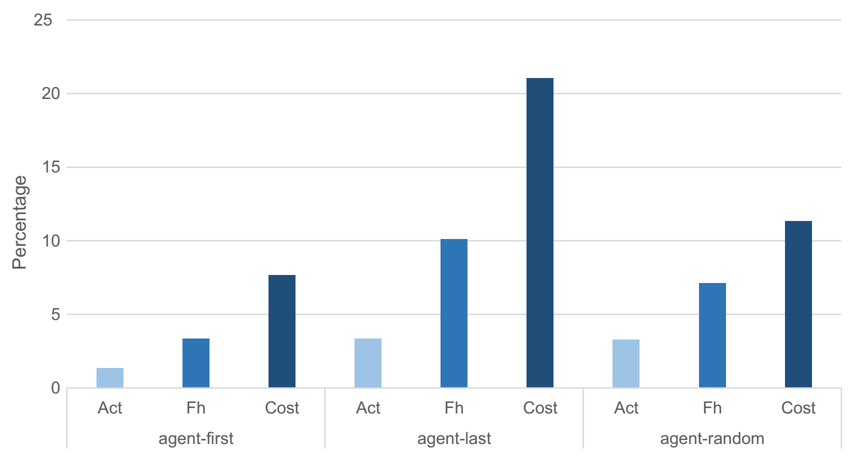

Fig. 4 Average increment in percentage of actions, finish time and cost of one agent in the IPG solution with respect to its best individual plan, when it goes first, last, or random in the BRPS order

According to Fig. 4, 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 first reaches 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% w.r.t. the number of actions of its best individual plan. On the other hand, the finish time 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 finish time 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 its plan first 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.

7 Conclusions and future work

Non-cooperative MAP can be seen as a problem that lies in between coalitional and adversarial planning, a field that has been hardly explored. The approach presented in this paper addresses this type of problems in which agents wish to make their interests prevail but also need to coordinate their strategic behavior with the others.

We defined a general-sum game 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. Since taxation schemes are applied to conflicts, agents are incentivized to avoid them. The inclusion of individual cost functions for the agents induces a strategic behavior and a more realistic representation of self-interested planning agents for real-world problems.

Regarding the potential version of the IPG, we showed 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 proved that any equilibrium is an IPG solution (conflict-free joint plan) in the absence of multi-symmetric unsolvable situations.

We analyzed 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 showed promising results towards PLS-completeness under some assumptions.

We experimentally compared our BRPS against BRP, one of the few available frameworks that present the same kind of strategic and cooperative behavior. The results show that BRPS clearly outperforms BRP in terms of computation time and quality of the PNE joint plans. The main advantages of BRPS with respect to BRP that we have shown through this work can be summarized as follows:

- As opposite to BRP, which follows a best-response scheme, BRPS is based on **better-response dynamics**. Better response (non-optimal plan) can be computed in polynomial time in many domains (i.e., transport domains without fuel restrictions), while synthesizing best responses (optimal plans) is always NP-complete [15].

- Unlike the centralized approach of BRP, BRPS relies on **fully-distributed** multi-agent technologies, which brings several advantages:
  - **preservation of the agents' privacy**: BRPS does not depend on a centralized planning entity with access to all the problem information. 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.
  - **self-sufficiency**: BRPS does not require an external input of a conflict-free joint plan. BRP, in contrast, must be initially fed with a conflict-free plan to run the best-response dynamics. Moreover, given the cooperative nature of this plan, the strategic behavior of the agents is not adequately reflected and agents are restricted to improve their plans on the basis of this conflict-free plan. However, in BRPS, agents compute a solution from scratch, calculating progressively an equilibrium solution that meets their interests.
  - **non-cooperative behavior**: in BRPS synergies among the agents' plans are not allowed. This ensures a non-cooperative behavior because agents cannot use the effects of the others' plans to achieve their own goals. Allowing synergies, as in BRP, yields a dependency between the agents' plans which must be maintained throughout in order to avoid conflicts. This exceedingly limits the strategic behavior of the agents as it is against the self-interested nature of the agents.
- Despite the use of complex multi-agent machinery, the better-response dynamics of our approach have been heavily optimized. As a result, BRPS scales up notably better than BRP, yielding significantly **lower computation times**.

Despite its superior performance against BRP, the results prove that BRPS is a costly procedure, which limits its applicability to tasks with a relatively low number of agents. Therefore, as future work, we intend to work on the optimization of the algorithms in order to improve the scalability of BRPS.

Acknowledgments This work is supported by the GLASS project TIN2014-55637-C2-2-R of the Spanish MINECO and the Prometeo project II/2013/019 funded by the Valencian Government.

References

1. Aghighi M, Bäckström C (2016) A multi-parameter complexity analysis of cost-optimal and net-benefit planning. In: Proceedings of the Twenty-Sixth International Conference on International Conference on Automated Planning and Scheduling. AAAI Press, London, pp 2–10
2. Bercher P, Mattmüller R (2008) A planning graph heuristic for forward-chaining adversarial planning. In: ECAI, vol 8, pp 921–922
3. Brafman RI, Domshlak C, Engel Y, Tennenholtz M (2009) Planning games. In: IJCAI 2009, Proceedings of the 21st international joint conference on artificial intelligence, pp 73–78
4. Bylander T (1994) The computational complexity of propositional strips planning. Artif Intell 69(1):165–204
5. Chen X, Deng X (2006) Settling the complexity of two-player nash equilibrium. In: 47th annual IEEE symposium on foundations of computer science, 2006. FOCS'06. IEEE, pp 261–272
6. Chien S, Sinclair A (2011) Convergence to approximate nash equilibria in congestion games. Games and Economic Behavior 71(2):315–327
7. de Cote EM, Chapman A, Sykulski AM, Jennings N (2010) Automated planning in repeated adversarial games. In: 26th conference on uncertainty in artificial intelligence (UAI 2010), pp 376–383
8. Dunne PE, Kraus S, Manisterski E, Wooldridge M (2010) Solving coalitional resource games. Artif Intell 174(1):20–50
9. Fabrikant A, Papadimitriou C, Talwar K (2004) The complexity of pure nash equilibria. In: Proceedings of the thirty-sixth annual ACM symposium on theory of computing, STOC '04, pp 604–612
10. Friedman JW, Mezzetti C (2001) Learning in games by random sampling. J Econ Theory 98(1):55–84
11. Ghallab M, Nau D, Traverso P (2004) Automated planning: theory & practice. Elsevier
12. Goemans M, Mirrokni V, Vetta A (2005) Sink equilibria and convergence. In: Proceedings of the 46th annual IEEE symposium on foundations of computer science, FOCS '05, pp 142–154
13. Hadad M, Kraus S, Hartman IBA, Rosenfeld A (2013) Group planning with time constraints. Ann Math Artif Intell 69(3):243–291
14. Hart S, Mansour Y (2010) How long to equilibrium? the communication complexity of uncoupled equilibrium procedures. Games and Economic Behavior 69(1):107–126
15. Helmert M (2003) Complexity results for standard benchmark domains in planning. Artif Intell 143(2):219–262
16. Helmert M (2006) The fast downward planning system. J Artif Intell Res 26(1):191–246
17. Jennings N, Faratin P, Lomuscio A, Parsons S, Wooldrige M, Sierra C (2001) Automated negotiation: prospects, methods and challenges. Group Decis Negot 10(2):199–215
18. Johnson DS, Papadimtriou CH, Yannakakis M (1988) How easy is local search? J Comput Syst Sci 37(1):79–100
19. Jonsson A, Rovatsos M (2011) Scaling up multiagent planning: a best-response approach. In: Proceedings of the 21st international conference on automated planning and scheduling, ICAPS
20. Jordán J, Onaindía E (2015) Game-theoretic approach for non-cooperative planning. In: 29th AAAI conference on artificial intelligence (AAAI-15), pp 1357–1363
21. McDermott D, Ghallab M, Howe A, Knoblock C, Ram A, Veloso M, Weld D, Wilkins D (1998) PDDL: the planning domain definition language. Yale Center for Computational Vision and Control, New Haven
22. Milchtaich I (1996) Congestion games with player-specific payoff functions. Games and Economic Behavior 13(1):111–124
23. Monderer D, Shapley LS (1996) Potential games. Games and Economic Behavior 14(1):124–143
24. Nigro N, Welch D, Peace J (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
25. Nisan N, Ronen A (2007) Computationally feasible vcg mechanisms. J Artif Intell Res 29(1):19–47
26. Nisan N, Roughgarden T, Tardos E, Vazirani VV (2007) Algorithmic game theory. Cambridge University Press, New York

27. Papadimitriou CH (1994) On the complexity of the parity argument and other inefficient proofs of existence. J Comput Syst Sci 48(3):498–532
28. Richter S, Westphal M (2010) The LAMA planner: guiding cost-based anytime planning with landmarks. J Artif Intell Res 39(1):127–177
29. Rosenthal RW (1973) A class of games possessing pure-strategy nash equilibria. Int J Game Theory 2(1):65–67
30. Shoham Y, Leyton-Brown K (2009) Multiagent systems: algorithmic, game-theoretic, and logical foundations. Cambridge University Press
31. Torreño A, Onaindia E, Sapena Ó (2014) A flexible coupling approach to multi-agent planning under incomplete information. Knowl Inf Syst 38(1):141–178
32. Torreño A, Onaindia E, Sapena Ó (2014) FMAP: distributed cooperative multi-agent planning. Appl Intell 41(2):606–626
33. Torreño A, Sapena Ó, Onaindia E (2015) Global heuristics for distributed cooperative multi-agent planning. In: ICAPS 2015. 25th international conference on automated planning and scheduling. AAAI Press, pp 225–233
34. Von Neumann J, Morgenstern O (2007) Theory of games and economic behavior. Princeton University Press
35. de Weerdt M, Bos A, Tonino H, Witteveen C (2003) A resource logic for multi-agent plan merging. Ann Math Artif Intell 37(1):93–130
36. Wooldridge M, Endriss U, Kraus S, Lang J (2013) Incentive engineering for boolean games. Artif Intell 195:418–439