Defeasible-Argumentation-based Multi-Agent Planning

Sergio Pajares Ferrando*, Eva Onaindia

Dpto. de Sistemas Informáticos y Computación
Universitat Politècnica de València
Camino de Vera s/n, E46022-Valencia (Spain)

Abstract

This paper presents a planning system that uses defeasible argumentation to reason about context information during the construction of a plan. The system is designed to operate in cooperative multi-agent environments where agents are endowed with planning and argumentation capabilities. Planning allows agents to contribute with actions to the construction of the plan, and argumentation is the mechanism that agents use to defend or attack the planning choices according to their beliefs. We present the formalization of the model and we provide a novel specification of the qualification problem. The multi-agent planning system, which is designed to be domain-independent, is evaluated with two planning tasks from the problem suites of the International Planning Competition. We compare our system with a non-argumentative planning framework and with a different approach of planning and argumentation. The results will show that our system obtains less costly and more robust solution plans.

Keywords: Defeasible Argumentation, Multi-Agent Planning, Multi-Agent Systems, Cooperation

1. Introduction

One common problem in Artificial Intelligence (AI) is to select the best course of action for an agent; i.e, reasoning about *what to do*. This problem has been primarily addressed from two standpoints: the knowledge or epistemological perspective, which puts the emphasis on the representation

*Corresponding author
Email addresses: spajaresf@gmail.com (Sergio Pajares Ferrando), onaindia@dsic.upv.es (Eva Onaindia)

of the world such that the solution of a problem follows from the representation; and the reasoning or heuristic perspective, mostly concerned with the information for solving the problem and reasoning on an abstract and formal representation of the world [23]. *Practical reasoning*, a research line primarily focused on the epistemological view, includes a great deal of epistemic reasoning, directed at determining what to believe [10, 19]. *Automated planning*, on the other hand, is concerned with the computational process for the selection and organization of the actions. Back in the 90's, Pollock concluded that since epistemic cognition is defeasible, a planning agent must be prepared to revise its plans as its defeasibly held beliefs change, and it may have to acquire more information through reasoning to solve a planning problem [35].

The mainstream in practical reasoning lies in the use of argumentation theory so as to extend the means-end reasoning in classical planning with presumptive justifications for the adoption of a particular action. The predominant approach in practical reasoning relies upon Dung's argumentation framework over beliefs [13] such as proposals for arguing about the desires an agent should adopt and the plans the agent will intend in order to achieve those desires [37]; the study of the goal deliberation process [20]; or the generation of consistent plans from a set of conflicting beliefs [3]. Building argumentation plans for negotiating conflict resolution at a planning stage is also an interesting application of argumentation in practical reasoning [25]. Some other works, however, follow the notion of argument scheme proposed by Walton [47] and present an approach in which arguments and conflicts are represented as argument schemes and critical questions, respectively [5]. This latter work has been one of the most popular approaches in practical reasoning, it has demonstrated its applicability in domains such as law, experimental economics or e-democracy [4, 12, 9] and it has also been exploited for the design of argumentation-based dialogues to support automated coordination in distributed planning [24], multi-agent deliberation dialogues [21] or the construction of joint plans [38].

Unlike argumentation-based approaches of practical reasoning, another line of investigation closer to *planning* also explores the relationships between classical planning and argumentation, building on a planning formalism and using argumentation to guide the reasoning process. A first step in this direction assumes that agent's deductions are not always certain information, but *plausible*, and the conclusions can be withdrawn when new pieces of knowledge are found; i.e., agents must use defeasible reasoning [34]. OSCAR is a goal-regression planner that essentially performs the same search of Partial-Order Planning (POP) but reasoning defeasibly about can-

didate plans at the end of the planning process [36]. In OSCAR, the search of a plan is also done defeasibly, thus enabling to reason about the impact of unexpected environmental conditions on the solution plans as well as selecting the plan which is less likely to fail at execution time. In the same line, another pioneer work presents a formal model of plans based on a defeasible argument system that is able to suggest aspects of a plan, criticize and revise the plan [14]. Both of these investigations, considered as the first steps towards building an *argumentation-based planning* system, have close similarities to the works on plan modification and replanning but rather than enforcing the planner to resort to replanning in light of new information, they consider planning within the context of a general defeasible reasoning system.

More recently, Simari et al presented a defeasible argumentation framework for the definition of actions and the combination of these actions into plans [40]. This work lays the foundations of an argumentation-based formalism for constructing plans [17] by using Defeasible Logic Programming (DeLP) [15], a formalism that agents use to represent and deal with incomplete and contradictory information in dynamic domains. The formalism presented in [17], which we will refer to as DeLP-POP in the following, describes how the traditional POP algorithm is extended to consider arguments as planning steps.

Subsequently, further investigations on argumentation-based planning focused on the application of argument-based systems to Multi-Agent Planning (MAP). An argumentation-based dialogue protocol that enables agents to discuss candidate plans and reach agreements was proposed in [8, 7]. In this work, candidate plans of the agents are generated by an external single-agent planner and the protocol is used for reasoning about the contradictory planning beliefs in the candidate plans and select a valid solution plan. Agents in [8, 7] use argumentation to defend or attack the candidate plans put forward by other agents, but not for cooperatively building a plan contributed by multiple agents. Another interesting work that exploits the benefits of using argumentation in MAP emphasizes the utilization of argumentation to solve conflicts between sub-plans of different agents by means of deliberative dialogues based on argumentation schemes [43, 44]. Conflicts may be caused by concurrent actions, plan constraints or norms that the agents must adhere to, and argumentation is used to analyze the conflicts that arise when several sub-plans of different agents are to be merged. Likewise, in this approach argumentation is not used for building a plan -this is accomplished by an external planner-, but for arguing at end of the planning generation. A different approach that also makes use of argumen-

tation schemes proposes structured argumentative dialogues to coordinate plan-related tasks [24]. In this proposal agents coordinate their beliefs and intentions using a dialogue game based on an argumentation scheme and its critical questions. Authors propose a strategy to choose relevant questions so as to improve the efficiency of dialogues and they empirically prove the benefits of the approach in identifying the points of disagreement to come to an agreement on the best plan.

On the other hand, the first formal extension of DeLP-POP to a multi-agent context wherein agents are assumed to have planning and argumentation capabilities is presented in [31]. Specifically, this work proposes a formal dialogue for an incremental argumentative plan search, by which agents exchange plan proposals and arguments for and against such proposals. To the best of our knowledge, this work represents the first attempt to use an argumentation-based MAP mechanism for a cooperative construction of plans. Subsequently, the works in [30, 28, 29] present the evaluation of the formal approach in some practical domains like a transit journey planning service and ambient intelligent applications.

In this paper, we present the formalization of Q-DeLP-POP[1] and its extension to a MAP environment (Q-DeLP-MAP), an argumentation-based MAP system that elaborates on two previous contributions: an initial formalization of a multi-agent argumentative planning model in the framework of DeLP-POP [31] and a preliminary implementation of such argumentative MAP model in a domain of ambient intelligent applications [28, 29]. The former version of the argumentation model was able to deal with rich argumentative representations but exhibited a limited planning capability. Q-DeLP-MAP, however, greatly outperforms the previous system by exploiting, among other things, the reuse of argumentative dialogues during the construction of the search tree, which allows us to tackle problems of the International Planning Competitions (IPC)[2]. Additionally, Q-DeLP-POP provides a more sophisticated specification of the *qualification problem* in planning, defining novel relationships between argument steps and action steps of a plan. Overall, the aim of this paper is to put together and exploit the investigations carried out in [31] and [29] in order to come up with a domain-independent, fully integrated and operative argumentation-based MAP model.

This paper is organized as follows. In sub-sections 1.1 and 1.2, we present

[1] *Q* stands for *Qualification problem* [18].
[2] http://ipc.icaps-conference.org/

the objectives of this work and we highlight the contributions of Q-DeLP-MAP with respect to former investigations, respectively. Section 2 summarizes the main foundations in which this work is based on. Section 3 presents in detail the components of our defeasible-argumentation-based planning framework. Section 4 presents the multi-agent protocol for our planning framework. Next, the experiments carried out to validate the framework are described and analyzed in section 5. Finally, we conclude and present some directions for future work.

1.1. Objectives and Problem Description

This work revolves around the problem of **Multi-Agent Planning** and, more specifically, on cooperative planning, wherein multiple entities (agents) work cooperatively in order to build a plan that solves a common goal. Under this paradigm, agents are interpreted as entities that have different planning capabilities (specialized agents) or entities that are geographically distributed and so they do not have access to all the data of the world. These different world views and planning capabilities define the agents' vision of the planning task.

In classical approaches to MAP, agents only work with the information defined in the planning task: the facts that describe the initial state of the world and the actions, which represent the agent capabilities and model the dynamics and causal relationships of the planning task. Agents apply a search process over the initial state and infer new facts that represent the evolution of the world through the application of the actions. Hence, classical planning approaches assume that the planner (agent) begins with all the relevant, accurate and necessary knowledge for solving the task. However, in complex and dynamic environments, it is required to represent and manage the *know-how knowledge* of a planning agent; i.e., information other than the causal inferences drawn from the application of the actions of the planning task. Our principal objective is thereby extending a traditional MAP task so that the know-how knowledge of the agents is considered when building a plan. This new source of information, which stems from the expertise and beliefs of the agents but cannot be regarded as a universal truth, will be modeled as **defeasible knowledge** which is susceptible of change in light of new information provided by other agents.

The goal of planning is choosing the proper actions to achieve the task goals according to some optimization criteria. This is done by using the factual information and known abilities of the agents. The purpose of introducing defeasible knowledge in a planning task is to account for all the

external conditions that might potentially affect the execution of an action in the real world beyond the conditions defined in the causal model. Thus, the agents' defeasible knowledge will be used to conduct the planning process towards a successful plan execution through a **multi-agent argumentation model**. Planning agents will use their know-how knowledge and beliefs of the world to argue about the context in which an action will be executed and to prevent the action from a potential execution failure. In some domains, it is crucial to obtain robust plans; i.e., plans that are executable despite the uncertainty in the execution environment. For example, consider a situation in which an ambulance needs to be sent to a place in an emergency situation. Traditional MAP approaches are not able to take into account the contextual conditions that might affect the execution of the action, such as the traffic conditions when driving the ambulance, because this information is not expressed in the causal theory. Consequently, the plan might possibly fail during its execution in the real world.

Our argumentation model draws upon a defeasible knowledge reasoner that allows agents to argue about the contextual conditions that could prevent an action of a plan from being successfully executed. From the planning standpoint, the goal is to build plans consistent with the causal theory that defines the dynamics of the world. From the argumentation perspective, the goal is to obtain plans consistent with the agents beliefs so as to increase the likelihood of achieving a robust plan. Thereby, the quality of the plans will be assessed according to the usual planning quality criteria (number of actions and plan duration) as well as by their level of robustness (minimization of the risk of unexpected action failures).

The operational framework that we present in this paper requires to design and build multi-agent dialogues so that each step of the construction of a plan can be discussed among agents. Specifically, we will design a dialogue mechanism by which agents exchange arguments about the conditions that might affect the feasibility of an action in the real world according to their know-how knowledge and beliefs. Therefore, agents will not only be equipped with planning capabilities, but also with argumentation abilities.

Our goal is to obtain more robust plans than the plans returned by traditional MAP systems (without argumentation); i.e., solutions that demonstrate that incorporating the know-how knowledge of the agents in the form of defeasible knowledge allows us to obtain more robust executable plans. For this purpose, agents will instantiate argumentative dialogues during planning to reason about each step comprised in the plan.

Finally, we aim for a domain-independent argumentation-based MAP model, fully integrated and operative. This means that we have not only

to design and implement the model but also test it in different applications domains. Specifically, the framework will be validated with problems from the IPC benchmarks. We will carry out several experiments considering various levels of difficulty of the planning and argumentation tasks.

1.2. Contributions of Q-DeLP-MAP

As commented above, Q-DeLP-MAP relies upon former investigations on argumentation and multi-agent planning. Q-DeLP-MAP represents the evolution of an approach that started by designing the principles and theoretical foundations underpinning a multi-agent extension of DeLP-POP up to the currently novel version, which is able to tackle general-purpose planning problems. Table 1 showcases the evolution of Q-DeLP-MAP through the various contributions along with the contribution of each approach.

The beginning of this work was a joint collaboration with other researchers where we established the theoretical foundations of a multi-agent extension of DeLP-POP [31]. In this work, the dialogues instantiate an $A^\star$ search algorithm that agents use to find a provably optimal solution according to their knowledge. MAPA defines the architecture and protocols for a cooperative distributed planning system based on defeasible reasoning [30]. The stages of the MAPA planning protocol are inherited by Q-DeLP-MAP but the work in [30] only shows a simple illustrative example on a *Transit Journey Planning Service*. Subsequently, we opted for creating an ad-hoc version of the planning-argumentation framework for Ambient Intelligence (AmI) Applications [28]. DeLP-MAP-POP featured rather limited planning capabilities but a rich argumentative model to represent AmI applications. The last step of this ad-hoc framework was CAMAP [29], which notably improved the performance of DeLP-MAP-POP. The novelty of CAMAP lied in the exploitation of context-aware information, particularly on the field of health-care problems. CAMAP was able to solve medium-size instances of homecare applications with up to 24 actions and 150 defeasible rules.

The motivation of Q-DeLP-MAP is to come up with a general-purpose planning and argumentation approach. Q-DeLP-MAP enhances the representation and management of the qualification problem, it proposes a separation of the planning and argumentation conflicts into *threats* and *attacks*, respectively, and includes a Case-Based Reasoning module that significantly improves the performance of the argumentation process. All these features together allow us to tackle medium-size domain-independent planning problems in a variety of domains like space applications, transport or logistics. To the best of our knowledge, Q-DeLP-MAP is the first domain-independent

Approach	**Main Contribution**	**Applicability**	**Performance**
Multi-Agent extension of DeLP-POP [31]	Formalization of multi-agent DeLP-POP	—	—
MAPA [30]	Planning and Argumentation architecture	Case study: Transit Journey Planning Service	—
DeLP-MAP-POP [28]	Preliminary version of an operational framework for [31] and [30]	Ambient Intelligence applications	Limited planning capabilities, rich argumentative representations (small instances of AmI applications)
CAMAP [29]	Refined version of DeLP-MAP-POP: Context-aware MAP model based upon an argumentation-based defeasible logic	Ambient Intelligence applications (healthcare)	Medium-size instances of AmI applications (24 actions, 150 defeasible rules)
Q-DeLP-MAP	Qualification problem; Planning and argumentation conflicts; Case-Based Reasoning	Domain-independent planning problems (space applications, transport, logistics)	Medium-size planning problems (500 actions, 180 defeasible rules)

Table 1: Evolution of prior works and contribution of Q-DeLP-MAP

argumentation-based MAP approach that is capable of solving various problems from the IPC.

2. Preliminary notions

In this section, we summarize the basic notions that will be used throughout this document.

2.1. DeLP*: a framework for defeasible argumentation*

Defeasible reasoning is a process where tentative conclusions are obtained from uncertain or incomplete information and so they may no longer be valid after new information becomes available [33]. Defeasible Logic Programming (DeLP) is a popular formalism to make context inferences by using *defeasible argumentation*, a particular type of defeasible reasoning applied to single-agent contexts [15].

In DeLP, the agent's knowledge base is a pair $T = (\Psi, \Delta)$, where Ψ is a consistent set of facts (literals) and Δ is the set of defeasible rules of the form $\delta = \ell \multimap \ell_0, \ldots, \ell_n$, being ℓ the head and $\ell_0, \ldots, \ell_n$ the body of Δ, respectively. The non-empty set of literals $\ell_0, \ldots, \ell_n$ provides a (defeasible) reason for ℓ to be *warranted*[3]. Given T, a *defeasible* derivation of a literal ℓ from T implies that there may exist information in contradiction with ℓ that will prevent the acceptance of ℓ as a valid conclusion. In general, the set of derivable literals in T will not be consistent and more than one defeasible derivation for a literal ℓ may be found as well as more than one defeasible derivation against it.

Defeasible argumentation is a form of defeasible reasoning that emphasizes the notion of an argument. An argument $\mathcal{A}$ for a literal ℓ of $T = (\Psi, \Delta)$ is a subset of defeasible rules, $\mathsf{rul}(\mathcal{A}) \subseteq \Delta$, such that (i) ℓ is derivable from $\Psi \cup \mathsf{rul}(\mathcal{A})$; (ii) the set $\Psi \cup \mathsf{rul}(\mathcal{A})$ is non-contradictory; and, (iii) $\mathsf{rul}(\mathcal{A})$ is a minimal subset of Δ satisfying (i) and (ii) [15]. Figure 1 shows an argument $\mathcal{A}$ for a literal ℓ, where [31]:

1) $\mathsf{rul}(\mathcal{A}) = \{\delta_0, \delta_1\}$, $\delta_0 = \ell \prec \{p_0, p_1\}$ and $\delta_1 = p_1 \prec \{q_0, q_1, q_2\}$
2) $\mathsf{base}(\mathcal{A}) = \mathsf{body}(\mathsf{rul}(\mathcal{A})) \setminus \mathsf{head}(\mathsf{rul}(\mathcal{A}))$; i.e, $\mathsf{base}(\mathcal{A}) = \{p_0, q_0, q_1, q_2\}$
3) $\mathsf{concl}(\mathcal{A}) = \mathsf{head}(\mathsf{rul}(\mathcal{A})) \setminus \mathsf{body}(\mathsf{rul}(\mathcal{A}))$; i.e, $\mathsf{concl}(\mathcal{A}) = \ell$

The existence of $\mathcal{A}$ for ℓ ($\mathsf{concl}(\mathcal{A}) = \ell$), given that $\mathsf{base}(\mathcal{A}) \subseteq \Psi$, still does not suffice for ℓ being warranted; we must guarantee that $\mathcal{A}$ is not

[3]Strict rules introduced in [39, 15] have not been considered in planning (see [17]).

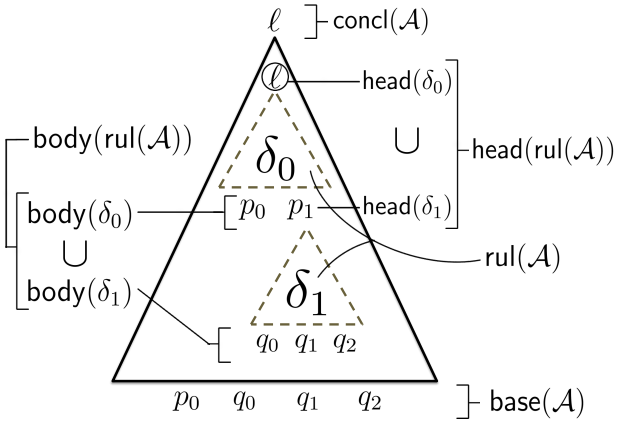

Figure 1: An argument $\mathcal{A}$ for l composed of two rules δ_0, $\delta 1 \in \mathsf{rul}(\mathcal{A})$.

defeated by some other argument. More specifically, given two arguments $\mathcal{A}_0$ and $\mathcal{A}_1$, where $\{\mathsf{base}(\mathcal{A}_0) \cup \mathsf{base}(\mathcal{A}_1)\} \subseteq \Psi$, we say that $\mathcal{A}_1$ attacks $\mathcal{A}_0$ if the conclusion of $\mathcal{A}_1$ contradicts some literal derived in $\mathcal{A}_0$, that is, if $\overline{\mathsf{concl}(\mathcal{A}_1)}$[4] $\in \mathsf{head}(\mathsf{rul}(\mathcal{A}_0))$. The attack relation may roughly be seen as symmetric, in the sense that each attacked argument $\mathcal{A}_0$ contains a sub-argument $\mathcal{A}_{0'}$ attacking $\mathcal{A}_1$. We use the same formal criterion of *specificity* of [15] for deciding the contending argument that prevails in an attack: the argument that is based on more information content or with less use of rules (more direct) is a *proper defeater* or a preferable argument. If two contending arguments are not comparable in these terms, they are a *blocking defeater* to each other[5]. A counter-argument $\mathcal{A}_1$ will either attack $\mathsf{concl}(\mathcal{A})$ or an inner point of $\mathcal{A}$; i.e., $\mathsf{body}(\mathsf{rul}(\mathcal{A})) \cup \mathsf{head}(\mathsf{rul}(\mathcal{A})) \setminus \mathsf{concl}(\mathcal{A})$.

Given an argument $\mathcal{A}_0$ for ℓ, an *argumentation line* $\Lambda = [\mathcal{A}_0, \ldots, \mathcal{A}_n]$ is a sequence of arguments constructible from (Ψ, Δ), where each argument $\mathcal{A}_{k+1}$ is a defeater for its predecessor $\mathcal{A}_k$. Arguments supporting (resp. interfering with) $\mathcal{A}_0$ are arguments of the form $\mathcal{A}_{2n}$ (resp. $\mathcal{A}_{2n+1}$), they must form a consistent set, and no sub-argument $\mathcal{A}'$ of an argument $\mathcal{A}_m \in \Lambda$ may appear later in Λ (i.e., it cannot be the case of $\mathcal{A}' = \mathcal{A}_{m'}$ with $m' > m$) [15, 17].

Given a root argument $\mathcal{A}$, the union of its argumentation lines gives rise to a tree-like structure, the *dialectical tree* for $\mathcal{A}$, denoted as $\mathcal{T}_{\mathcal{A}}(\Psi, \Delta)$. Figure 2 shows an example of two dialectical trees. In order to check whether the argument of the root node $(\mathcal{A})$ is defeated or undefeated, the following

[4]The set of negated literals of $\mathsf{concl}(\mathcal{A}_1)$.

[5]Alternatively, other comparison criteria between arguments could be defined along with a defeat relation induced from the selected preference criterion. See [39] for details.

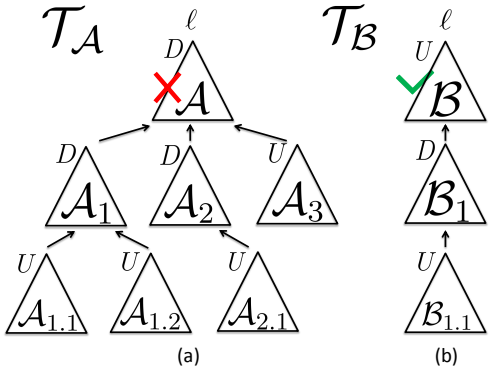

Figure 2: Computing warrancy for ℓ: (a) $\mathcal{T}_\mathcal{A}$: $\mathcal{A}$ is a defeated argument, and (b) $\mathcal{T}_\mathcal{B}$: $\mathcal{B}$ is an undefeated argument.

procedure on the *dialectical tree* is applied [15]: label with a U (for *undefeated*) each terminal node in the tree (i.e. each argument with no defeaters at all). Then, in a bottom-up fashion, label a node with:

$$\begin{cases} U & \text{if each of its successors is labeled with a } D \\ D & \text{(for } \textit{defeated}\text{) otherwise} \end{cases}$$

The application of this procedure returns that $\mathcal{A}$ is a defeated argument in Figure 2 (a), and that $\mathcal{B}$ is an undefeated argument in Figure 2 (b).

2.2. POP*: Partial Order Planning*

Partial-Order Planning (POP) draws upon the least commitment principle [32, 48], which delays commitment of action orderings until a decision is necessary to solve some inconsistency. In POP, a plan is represented as a set of actions and a set of ordering constraints defining a partial order between the actions. The POP paradigm is a flexible mechanism to deal with the individual plans of different agents and combine them into a single joint plan. Since our goal is the integration of planning and defeasible argumentation in a multi-agent context, we adopt POP as the planning approach of the agents.

A planning task is defined as a tuple $\mathbb{M} = (\Psi, A, G)$, where Ψ is the set of facts that represent the initial state of the planning task, A is the set of actions and, G is the set of goal literals. An action $\alpha = \langle \mathsf{pre}(\alpha), \mathsf{eff}(\alpha) \rangle$ is a set of preconditions (for α to be applicable) and effects.

A POP plan Π, or simply a plan Π, is a 3-tuple $\langle A(\Pi), CL(\Pi), OC(\Pi) \rangle$, where $A(\Pi)$ is the set of action steps in Π, and $CL(\Pi)$ and $OC(\Pi)$ are the set of causal links and ordering constraints, respectively, between the action steps of $A(\Pi)$. A *causal link* between two actions steps, α_i and α_j,

is denoted as $(\alpha_i, \ell, \alpha_j) \in CL(\Pi)$, meaning that $\ell \in \mathsf{pre}(\alpha_j)$ is planned to be supported by $\ell \in \mathsf{eff}(\alpha_i)$. In addition, for a certain pair of action steps α_i and α_j, α_i may precede α_j or viceversa. Such a relationship is called *ordering constraint* and is denoted as $\alpha_i \prec_\Pi \alpha_j$ (or $\alpha_j \prec_\Pi \alpha_i$) $\in OC(\Pi)$. A causal link between two actions defines an implicitly ordering constraint between both. It is important to note that in a partially ordered set not every pair of actions need to be ordered: for a given pair of action steps, it may be that neither action step precedes the other in the plan. In POP, Ψ and G are encoded as dummy actions $\alpha_\Psi \prec_\Pi \alpha_G$ with $\mathsf{eff}(\alpha_\Psi) = \Psi$, $\mathsf{pre}(\alpha_G) = G$ and $\mathsf{pre}(\alpha_\Psi) = \mathsf{eff}(\alpha_G) = \emptyset$. Finally, the set of unsupported preconditions of the action steps in Π is called *open goals*, and it is denoted as $\mathsf{goals}(\Pi)$.

The elements of $A(\Pi)$ and $CL(\Pi)$ may cause the appearance of *threats* in a plan. Let's suppose a causal link $(\alpha_i, \ell, \alpha_j)$; and an action β, such that $\overline{l} \in \mathsf{eff}(\beta)$, which is unordered with respect to α_i and α_j. A threat, denoted by $(\beta, (\alpha_i, \ell, \alpha_j))$, means that the interfering or *threatening* action β would invalidate the causal link if β is eventually ordered between the two actions of the causal link, α_i and α_j. When detected, threats are to be solved by some threat resolution step:

- *promotion*: the action step α_j is ordered before β establishing the ordering constraint $\alpha_j \prec_\Pi \beta$; this results in the sequence of action steps $\alpha_i \prec_\Pi \alpha_j \prec_\Pi \beta$
- *demotion*: the action step α_i is ordered after β establishing the ordering constraint $\beta \prec_\Pi \alpha_i$; this results in the sequence of action steps $\beta \prec_\Pi \alpha_i \prec_\Pi \alpha_j$

The set of all the threats in a plan Π is labeled as $\mathsf{threats}(\Pi)$, and initially, $\mathsf{threats}(\Pi) = 0$. The set of *flaws* of a plan Π includes $\mathsf{threats}(\Pi)$ and $\mathsf{goals}(\Pi)$, and is denoted as $\mathsf{flaws}(\Pi)$. A plan Π is solution if $\mathsf{flaws}(\Pi) = \emptyset$; i.e, if Π is a threat-free plan and all the goals in G are achieved through a causal link. If Π is a solution plan, then $A(\Pi)$ applied over the initial state Ψ leads to a problem state in which the goals of the task, G, hold.

2.3. DeLP-POP*: A first extension of* POP *with* DeLP

DeLP-POP is a theoretical extension of POP with DeLP-style argumentation [17] in which both arguments and actions are used to resolve goals and threats. DeLP-POP distinguishes between actions steps and argument steps. Actions are intended to express the physics of a domain so the effects of an

action reflect the changes that will be produced in the world when the action is executed. However, argument steps represent the conclusion inferred by an agent according to its local know-how knowledge and partial view of the world. The novelty of DeLP-POP is that arguments can be introduced in the plan to support action preconditions. Consequently, the conclusion derived through an argument may be invalidated if another agent puts forward an opposite conclusion.

A planning task $\mathbb{M}$ is extended in DeLP-POP as a tuple (Ψ, Δ, A, G), where the new element Δ contains defeasible rules that may apply in any of the world states that result from the execution of the actions of the plan. An agent will use the defeasible rules in Δ for building arguments during the planning search. Let ℓ be an open goal, motivated by some step $\beta \in A(\Pi)$ or $\mathcal{A} \subseteq \Delta$; i.e. $\ell \in \mathsf{pre}(\beta)$ or $\ell \in \mathsf{base}(\mathcal{A})$. If goal ℓ is planned to be enforced by an action α, this is encoded as a *causal link* of Π and included in the set $CL(\Pi)$: $(\alpha, \ell, \kappa) \in CL(\Pi)$, with $\kappa = \beta$ or $\kappa = \mathcal{A}$. If $\ell \in \mathsf{pre}(\beta)$ and it is enforced by an argument $\mathcal{B}$, this is encoded as a *support link* of Π, included in a set labeled $SL(\Pi)$: $(\mathcal{B}, \ell, \beta) \in SL(\Pi)$, where $\mathcal{B} \subseteq \Delta$.

A plan Π for a DeLP-POP task $\mathbb{M}$, is a 5-tuple $\langle A(\Pi)$, $AR(\Pi)$, $CL(\Pi)$, $SL(\Pi)$, $OC(\Pi)\rangle$, where $A(\Pi)$, $CL(\Pi)$ and $OC(\Pi)$ are the same elements of a POP plan (see section 2.2); $AR(\Pi)$ represents the set of argument steps or, more particularly, the utilization of the defeasible rules in Δ within Π; and $SL(\Pi)$ is the set of support links.

In DeLP-POP, arguments are not only introduced to intentionally support the precondition of an action but they are also used to defeat or defend other arguments in the plan. When actions and arguments are combined in a partial-order plan, new types of threats arise [17, 16] (more details on the DeLP-POP threats are exposed in section 3.2.1).

3. Components of Q-DeLP-POP

In this section, we present a thorough formalization of Q-DeLP-POP, a novel argumentative planning framework that extends DeLP-POP [17] for dealing with the *qualification problem* [18]. The adopted solution to the *qualification problem* leads to redefine some of the components of the DeLP-POP framework [17] as well as to introduce new ones. The motivation behind Q-DeLP-POP stems from the fact that in natural environments the successful execution of actions can never be predicted with absolute certainty. Unexpected circumstances, albeit unlikely, may at any time prevent an agent from performing the intended actions [42].

The qualification problem is concerned with the impossibility of listing all the preconditions required for a real world action to have its intended ef-

fect. Within the planning community, the default choice to the qualification problem is to assume away the numerous possible unexpected circumstances that may prevent an action from being executed in the real world and resort to replanning in case a plan turns out not to be executable due to a non-anticipated condition. Some approaches assume an action is computed without considering the qualifying domain constraints and the action is taken to qualify if and only if any of the constraints is violated after the computation is complete [18]. Other methods cope with the qualification problem by respecting causality when minimizing anomalous models, which requires a prior definition of abnormal qualifications and anomalous models [42]. Our method to overcome the qualification problem during the construction of a plan relies on the local know-how knowledge of the agent about anomalous situations. Rather than verifying all unusual situations in the preconditions of the actions, agents put forward an action to the plan in the form of an argument so that any other agent possessing information about a tentative anomalous situation that might prevent the action from being executed launches an attack against the argument representing the action. This new semantic representation will be shown hereinafter.

3.1. Action-Argument Steps

Q-DeLP-POP provides a novel representation mechanism of actions, through which agents are able to attack an action step of the plan if they hold information about an environmental condition that could potentially prevent the action from having its intended effects. Given an action α that is to be inserted in a plan Π as an action step of $A(\Pi)$, we generate $\mathsf{eff}(\alpha)$ through a defeasible derivation and encode such a derivation as an argument of Π. For this purpose, Q-DeLP-POP replaces the notion of action step (elements in $A(\Pi)$) by a new compound entity called **action-argument step**, whereas it keeps the notion of argument step ($AR(\Pi)$) as defined in DeLP-POP.

Definition 1. **[Action-Argument Step]**. *Let α be an action with $\mathsf{eff}(\alpha) = \ell$ to be inserted in a plan Π. In* Q-DeLP-POP*, α is inserted in Π as a pair action-argument $\gamma = \langle \alpha', \mathcal{A} \rangle$. Particularly:*

- $\mathsf{eff}(\alpha') = \mu_{\alpha'}$*, where $\mu_{\alpha'}$ is a fictitious and irrevocable effect used to denote the actual execution of action α'.*
- $\mathsf{rul}(\mathcal{A}) = \ell \prec \mu_{\alpha'}$*, where* $\mathsf{base}(\mathcal{A}) = \mu_{\alpha'}$ *and* $\mathsf{concl}(\mathcal{A}) = \mathsf{eff}(\alpha)$*.*
- $\mathsf{pre}(\gamma) = \mathsf{pre}(\alpha') = \mathsf{pre}(\alpha)$*.*
- $\mathsf{eff}(\gamma) = \mathsf{concl}(\mathcal{A}) = \mathsf{eff}(\alpha)$*.*

Action steps are all replaced in Q-DeLP-POP by action-argument steps. This leads to a redefinition of a plan Π as a 5-tuple $\langle AA(\Pi), AR(\Pi), CL(\Pi), SL(\Pi), OC(\Pi)\rangle$, where $AA(\Pi)$ represents the set of action-arguments in Π. We will call the set $AR(\Pi)$ *supporting arguments* so as to distinguish them from the arguments comprised in $AA(\Pi)$, which simply are a type of fictitious arguments artificially created to transform the effect of an action into a defeasible derivation.

The introduction of the compound entity action-argument step also implies to revisit the notions of *causal link* and *support link*. Specifically, let $\gamma_2 = \langle \alpha'_2, \mathcal{B}\rangle$ be the action-argument of an action α_2, which is represented on the right of Figure 3[6]; and let $\mathsf{pre}(\gamma_2) = \mathsf{pre}(\alpha'_2) = \ell$ be an open goal of γ_2; if ℓ is supported by the action-argument step $\gamma_1 = \langle \alpha'_1, \mathcal{A}\rangle$ of an action α_1, represented on the left of Figure 3, then Q-DeLP-POP introduces a *causal link* $(\gamma_1, \ell, \gamma_2) \in CL(\Pi)$, as shown in Figure 3.

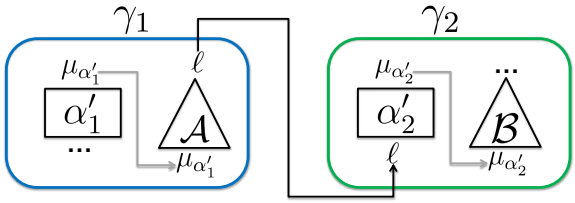

Figure 3: Example of causal link $(\gamma_1, \ell, \gamma_2) \in CL(\Pi)$.

Similarly to DeLP-POP, where a precondition of an action step can be supported by an argument step, we allow preconditions of action-argument steps to be likewise supported by supporting arguments. Specifically, let's consider the action α_2 in Figure 4 represented through the action-argument γ_2. Given $\ell \in \mathsf{pre}(\gamma_2)$, if ℓ is supported by an argument step $\mathcal{C}$, then Q-DeLP-POP introduces a support link $(\mathcal{C}, \ell, \gamma_2) \in SL(\Pi)$ and inserts $\mathcal{C}$ in the set $AR(\Pi)$. This is graphically represented in Figure 4 where the supporting argument $\mathcal{C}$ is used to satisfy $\mathsf{pre}(\gamma_2)$.

Finally, as it also occurs in DeLP-POP, a literal p that belongs to the base of an argument, say $p \in \mathsf{base}(\mathcal{C})$, can be supported by an action-argument $\gamma_3 = \langle \alpha'_3, \mathcal{D}\rangle$, thus introducing a *causal link* $(\gamma_3, p, \mathcal{C}) \in CL(\Pi)$, as shown on the left of Figure 4.

In case that an action α has more than one effect, i.e., $\mathsf{eff}(\alpha) = \{\ell_0, \ell_1, \ldots, \ell_n\}$, the corresponding action-argument step is denoted as a 2-tuple $\gamma = \langle \alpha, \{\mathcal{A}_0, \mathcal{A}_1, \ldots\}\rangle$, where $\mathsf{rul}(\mathcal{A}_0) = \ell_0 \multimap \mu_\alpha$, $\mathsf{rul}(\mathcal{A}_1) = \ell_1 \multimap \mu_\alpha$, and so on.

[6]Note that we represent the effects of α'_1 above the rectangle of action α'_1 and its preconditions below.

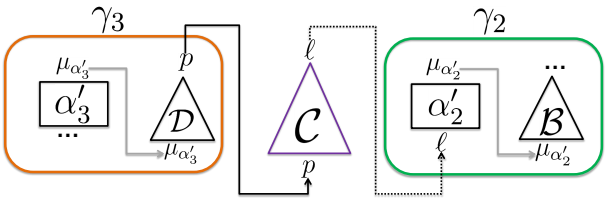

Figure 4: Example of support link $(\mathcal{C}, \ell, \gamma_2) \in SL(\Pi)$ and causal link $(\gamma_3, p, \mathcal{C}) \in CL(\Pi)$.

As we will see in section 3.2.2, representing the effects of an action through a defeasible derivation instead of a strict derivation as in DeLP-POP, allows agents to put forward arguments for or against the successful execution of the action and the achievement of its intended effects.

We say that a literal ℓ is an open goal in Π, denoted as $\ell \in \mathsf{goals}(\Pi)$, if $\exists \gamma \in AA(\Pi) \mid \ell \in \mathsf{pre}(\gamma)$ or $\exists \mathcal{A} \in AR(\Pi) \mid \ell \in \mathsf{base}(\mathcal{A})$, and it does not exist an element in $CL(\Pi)$ or $SL(\Pi)$ for the precondition of the action-argument or an element in $CL(\Pi)$ for the base of the argument, respectively, that supports ℓ.

3.2. Conflicting situations

When actions and arguments are involved in the construction of a plan, new types of conflicting situations are identified and need to be solved. Unlike DeLP-POP [17, 16], we distinguish two types of conflicting situations in a plan:

- *Threats*: they occur between two unordered steps of the plan such that if one is ordered before the other, the first one invalidates the application of the second one. Threats need to be solved in order to ensure the validity or correctness of the plan.
- *Attacks*: they occur when an agent holds some belief that contradicts the conclusions of a step of the plan. Attacks arise because of the contradictory information between the agents and they must be handled so as to have a greater guarantee of a successful plan execution.

While threats are used to validate the plan according to the physics expressed in the domain of the problem, attacks respond to the contextual or know-how information held by the agents (beliefs), which is not expressed in the physics of the domain. Hence, threats are aimed at checking the validity of the plan while attacks are aimed at checking the executability of the actions in the plan and the achievement of their intended effects. Next, we show an example for the ease of understanding.

Example 1. *In Figure 5(a), the action-argument step γ_1 represents the planning action 'fly plane apn1 from Munich to London'. The argument step $\mathcal{A}$ derives 'apn1 at Munich' according to the information provided by the air traffic control tower of Munich airport. The precondition of γ_1 is thus supported by the conclusion of $\mathcal{A}$. In this case, the conflicting situation occurs when a new action-argument, step γ_2, contradicts the support link $(\mathcal{A}$, apn1-at-Munich, $\gamma_1) \in SL(\Pi)$. This conflicting situation is similar to the classical notion of threat in* POP *but in this case γ_2 threatens a support link. On the other hand, Figure 5(b) shows two arguments: argument $\mathcal{B}_1$ denotes there are reasons to believe 'a volcanic and ash cloud' might happen and argument $\mathcal{B}_2$ denotes that the agent believes 'an airport strike might occur'. $\mathcal{B}_1$ and $\mathcal{B}_2$ are not part of the plan since they are not used to support any open goal of the plan, but they are a consequence of the beliefs of the agents, which may prevent γ_1 from achieving its intended effects. In other words, agents are saying that there are reasons to believe that these two unexpected circumstances ('a volcanic ash cloud' and 'airport strike') may occur and prevent the plane from reaching London. This conflicting situation is known in* Q-DeLP-POP *as an attack.*

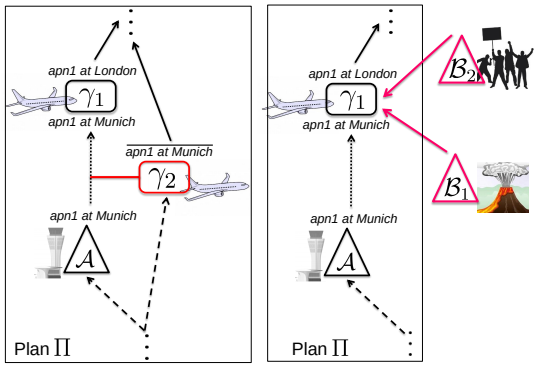

Figure 5: Example 1: (a) a threat in the plan; and, (b) attacks to the plan.

3.2.1. Threats

A threat happens when some step of the plan threatens the support provided by another step of the plan. Formally, we define a threat as a tuple $\langle k_3, (k_1, \ell, k_2)\rangle$, where k_1, k_2 and k_3 are action-argument steps or supporting arguments; k_3 is the threatening step which threatens the support (k_1, ℓ, k_2), being k_1 the step which provides some support to k_2. By keeping this uniform format for all types of threats, a neater threat classification as well as a simpler definition of the solutions can be defined.

Threats in Q-DeLP-POP are classified according to the type of support which is being threatened: a causal link, a support link or an internal support of an argument. Given a plan Π, and following the above threat definition $\langle k_3, (k_1, \ell, k_2)\rangle$, we define three types of threats.

Causal Link Threat (CLT). This threat occurs when a plan step threatens a causal link of the plan. Let (k_1, ℓ, k_2) be a causal link; then, k_1 is an element in $AA(\Pi)$, k_2 is an action-argument or a supporting argument of the plan and k_3 is always an action-argument step because k_3 is threatening a causal link of the plan. The causal link (k_1, ℓ, k_2) can be any of the causal links represented in Figures 3 and 4. In the first case, the CLT represents a classical POP threat. In the second case, k_2 is a supporting argument. In both cases, the threat is solved by applying demotion or promotion. Our CLTs are denoted as *action-action* and *action-base* threats in [16].

Support Link Threats (SLT). This threat occurs when a plan step threatens a support link of the plan. Let (k_1, ℓ, k_2) be a support link where $k_1 \in AR(\Pi)$ and $k_2 \in AA(\Pi)$. In this case, k_3, the threatening step, is an action-argument or a supporting argument. The former case is shown in Figure 6(a), where $k_1 = \mathcal{B}$, $k_2 = \gamma_2$ and $k_3 = \gamma_3$. This threat is solved by applying promotion ($\gamma_2 \prec_\Pi \gamma_3$). The case in which $k_3 \in AR(\Pi)$ is shown in Figure 6(b). The threatening step is $k_3 = \mathcal{C}$, which has been introduced in the plan to support the precondition of some other action (this is not graphically represented). Although the appearance of $\mathcal{C}$ in the plan can be interpreted as an interference with other steps of Π, this type of threat is not solved until base($\mathcal{C}$) is supported by an action-argument since ordering constraints are only established between action-arguments. This does not entail any drawback since the resolution of threats, as potential conflicts they are, can be postponed in the problem-solving process. The solution to this threat is to insert the ordering constraint ($\gamma_2 \prec_\Pi \gamma_5$). The threat in Figure 6(a) is denoted as an *action-argument* threat in DeLP-POP; and the threat in Figure 6(b) is included in the *argument-argument* threat definition of DeLP-POP.

Internal Support Threats (IST). This threat occurs when a plan step threatens an internal literal derived by some argument step of the plan. The two situations that arise as an IST are depicted in Figures 6(c) and 6(d). In both cases, $k_1 \in AR(\Pi)$, $k_2 \in AA(\Pi)$. The support (k_1, ℓ, k_2) is not of the form of a link but an internal literal n in the supporting argument $k_1 = \mathcal{B}$ that is necessary to derive the conclusion ℓ for $k_2 = \gamma_2$. In case 6(c), $k_3 = \gamma_3$ is the threatening step (n is an internal literal derived in the argument step $\mathcal{B}$), and the solution to this threat is to insert the ordering

constraint ($\gamma_2 \prec_\Pi \gamma_3$). In case 6(d), the solution also involves the application of promotion through the ordering constraint ($\gamma_2 \prec_\Pi \gamma_5$). The ISTs are included in the *action-argument* and *argument-argument* threats of DeLP-POP since this framework does not make any distinction in the type of support that is being threatened.

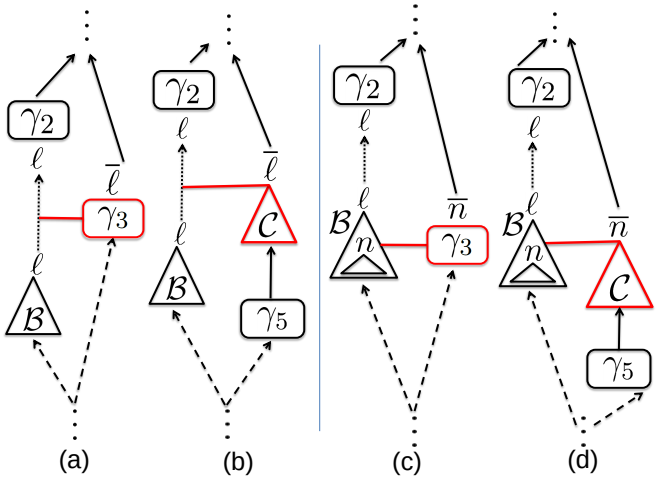

Figure 6: Examples of threats of type SLT and IST.

The work in [16] introduces one more type of threat called *action-assumption*, in which an action threatens the base of an argument even though the base is not warranted yet (this is the reason why it is so-called *assumption*). However, under our threat definition, a threat does not exist until a support is explicitly introduced in the plan; that is, until the base of the argument is warranted via action-arguments, in which case it would become a CLT and would be solved by promotion or demotion. It has been proved that many potential threats can actually be delayed until the end [41] and in practice many planners adopt threat deferral as a flaw selection strategy [48, 41]. The inclusion of supporting arguments in the threat machinery does not change this circumstance because the no-argument-supports-argument policy implies that any threatening or threatened supporting argument is involved in a threat as long as its base is supported by an action-argument.

3.2.2. Attacks

Attacks are conflicting situations that happen in a plan as a notice issued by an agent that some unexpected circumstances may affect the executability of an action in the plan. Attacks are based on the contextual information or beliefs of the agents and they are identified in order to provide a stronger

guarantee of success of the plan at execution time.

Formally, given a plan Π, we define an attack as a tuple $\langle \mathcal{A}_0, \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}, \Psi \rangle$, where:

- $\mathcal{A}_0$ is an argument of Π, either a supporting argument ($AR(\Pi)$) or a fictitious argument of an action-argument ($AA(\Pi)$).
- $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}$ is the set of attacking arguments such that $\mathcal{A}_1$ attacks $\mathcal{A}_0$, $\mathcal{A}_2$ attacks $\mathcal{A}_1$, and so on.
- Ψ is a set of literals that denote the activation context for the attacking arguments within the plan[7].

The first two elements of the tuple form an argumentation line [$\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{A}_2$, $\ldots \mathcal{A}_n$] as defined in section 2.3.

Example 2. *Let's suppose an emergency situation where the system demands a plan to move an ambulance from the hospital to the patient's home. An agent believes that the traffic congestion in the road that connects the hospital and the patient's home may prevent the ambulance from arriving in time. This contextual information about the traffic condition is very helpful to build a plan with guarantee of being successfully executed; that is, that the ambulance will be at the patient's home in time [28].*

Formally, given an agent whose knowledge base is $T = (\Psi, \Delta)$, it can build an argument $\mathcal{A}_1$ in T that attacks an argument $\mathcal{A}_0$ of a plan Π if the conclusion of $\mathcal{A}_1$ contradicts some literal derived in $\mathcal{A}_0$; that is, if $\overline{\mathsf{concl}(\mathcal{A}_1)} \in \mathsf{head}(\mathsf{rul}(\mathcal{A}_0))$ (see section 2.1). Particularly, in Example 2, the agent builds an argument $\mathcal{A}_1$ such that $\mathsf{concl}(\mathcal{A}_1) =$*'ambulance not on time'* contradicts $\mathsf{concl}(\mathcal{A}_0) =$*'ambulance on time'* given that both arguments use the same road between the hospital and the patient's home ($\mathsf{base}(\mathcal{A}_0) \cup \mathsf{base}(\mathcal{A}_1) \subseteq \Psi$) and the agent is aware of a congestion situation in such a road. This attack is represented as $\langle \mathcal{A}_0, \{\mathcal{A}_1\}, \Psi \rangle$ where: $\mathcal{A}_0$ is the argument of Π; $\mathcal{A}_1$ is the attacking argument; and, Ψ is the set of literals where $\mathsf{body}(\mathcal{A}_1)$ is warranted. Additionally, an argument $\mathcal{A}_2$ against $\mathcal{A}_1$ can be built in Example 2 if, for instance, an agent also holds an additional information of the existence of a fast track lane to avoid the traffic congestion. In this case, the argument $\mathcal{A}_2$ constructible in (Ψ, Δ) is a defeater of argument $\mathcal{A}_1$. The new tuple defining the attack would be $\langle \mathcal{A}_0, \{\mathcal{A}_1, \mathcal{A}_2\}, \Psi \rangle$ and [$\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{A}_2$] would be the argumentation line. Note that the attacking arguments $\mathcal{A}_1$

[7]The meaning of Ψ is the same as in the knowledge base $T = (\Psi, \Delta)$ of an agent but here it is referred to a particular context of the plan Π.

and $\mathcal{A}_2$ do not support any item in $\mathsf{goals}(\Pi)$ since they are merely agents' beliefs specifically used to attack or defend $\mathcal{A}_0$.

An initial attack of an argument $\mathcal{A}_1$ against an argument $\mathcal{A}_0$ of a plan Π is represented in Figures 7(a), 7(b), 7(c) and 7(d). The four figures feature the attack $\langle \mathcal{A}_0, \{\mathcal{A}_1\}, \Psi \rangle$, where $\mathcal{A}_0$ is the fictitious argument of γ_1 in Figures 7(a) and Figure 7(d), and a supporting argument ($\mathcal{A}_0 \in AR(\Pi)$) in Figures 7(b) and Figure 7(c). Particularly, in Figure 7(b), $\mathcal{A}_1$ is attacking $\ell \in \mathsf{concl}(\mathcal{A}_0)$; and, in Figure 7(c), $\mathcal{A}_1$ is attacking n, an internal literal of $\mathcal{A}_0$. This different attacking point to the argument $\mathcal{A}_0$ does not make any difference in the semantics of the attack. Although not graphically represented in Figure 7, an argument $\mathcal{A}_2$ could in turn attack $\mathcal{A}_1$, thus giving rise to the attack tuple $\langle \mathcal{A}_0, \{\mathcal{A}_1, \mathcal{A}_2\}, \Psi \rangle$.

It is important to note that all the attacking arguments $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}$ have to be activated in the same context Ψ of the plan Π; that is, $\mathsf{base}(\mathcal{A}_1)$, $\mathsf{base}(\mathcal{A}_2)$, ..., $\mathsf{base}(\mathcal{A}_n)$ have all to be warranted in Ψ, which in turn depends on the plan step to which argument $\mathcal{A}_0$ is giving support (for instance, in Figure 7, argument $\mathcal{A}_0$ is giving support to the action-argument γ_2). The activation of the attacking arguments is explained in detail in the next section.

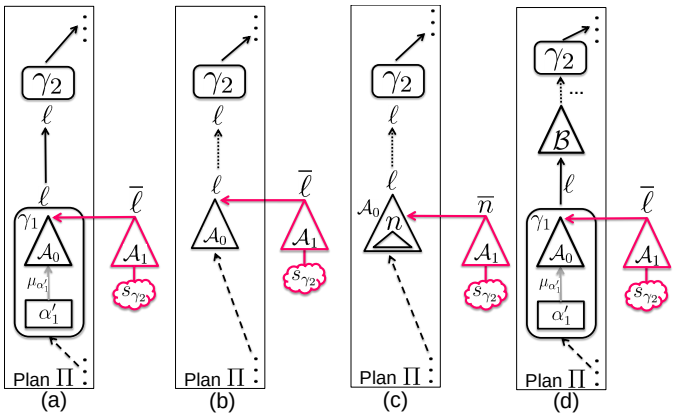

Figure 7: Examples of types of attacks.

Given a root argument $\mathcal{A}_0$, the union of its argumentation lines form a tree-like structure, the dialectical tree for $\mathcal{A}_0$, as explained in section 2.1. In order to evaluate the root argument, the same labeling procedure described in section 2.1 is applied. At the end, $\mathcal{A}_0$ will be labeled as U (undefeated) or D (defeated). If $\mathcal{A}_0$ is labeled as U, it means there is no evidence against the conclusions of $\mathcal{A}_0$. Otherwise, agents will consider that there are reasons to believe that conclusions of $\mathcal{A}_0$ might not be achieved.

3.3. Activation of Attacking Arguments

The two attacking arguments $\mathcal{A}_1$ and $\mathcal{A}_2$ of Example 2 must be constructible in (Ψ, Δ), meaning that the base of both arguments is to be warranted in Ψ. Therefore, we need to check that the base of the attacking arguments for or against an argument $\mathcal{A}_0$ of Π is warranted in the state of the plan step to which $\mathcal{A}_0$ is giving support. That is, an attack or defense to $\mathcal{A}_0$ affects the plan step being supported by $\mathcal{A}_0$ and, consequently, the arguments must be activated in the same context in which the corresponding plan step would be executed. In order to prevent an attacking argument from a false activation, we need to define a specific procedure to compute 'the state of a plan step'.

In state-based planning, a plan is a linear sequence of actions and the consistent state that holds before each action is known. However, a partial order plan Π is a set of actions whose execution ordering $\prec_\Pi$ is only partially specified, thus encoding multiple linear plans. Hence, POP does not explicitly represent state information associated to the actions in the plan.

Since states are not explicitly represented in POP, Q-DeLP-POP calculates, for each $\gamma \in AA(\Pi)$, the possibly inconsistent set of literals planned to occur before γ. Specifically, in [31], we presented the problem of identifying possible states in a partial plan with the notion of *proto-state*. The *proto-state* (also known as *Cutsets* in [26]) of an action-argument step γ can be seen as the set of literals that may hold before γ.

The set of *proto-states* of a plan must be continuously updated every time a step or an order constraint is inserted into the plan. As the plan search progresses, the *proto-states* will better match the final states that will result from the execution of the solution plan.

Definition 2. [Q-DeLP-POP **Proto-State**]. *Let* Π *be a plan for* $\mathbb{M}$*, and* $\gamma, \gamma', \gamma''', \ldots \in AA(\Pi)$*. In* Q-DeLP-POP*, the proto-state of an action-argument step* γ *in* Π*, labeled as* s_γ*, is comprised by*[8]*:*

$$s_\gamma = \{\ell \in \mathsf{Lit} \mid \exists \gamma' \in AA(\Pi), \ell \in \mathsf{eff}(\gamma') \text{ and } \prec_\Pi \cup \{\langle \gamma', \gamma \rangle\} \text{ is consistent, and } \forall \gamma'' \in AA(\Pi), \text{ if } \overline{\ell} \in \mathsf{eff}(\gamma'') \text{ then } \{\langle \gamma', \gamma'' \rangle, \langle \gamma'', \gamma \rangle\} \nsubseteq \mathsf{tc}(\prec_\Pi \cup \{\langle \gamma', \gamma \rangle\})\}$$

Figure 8 shows an example of a partial plan with five action-argument steps. The *proto-state* of γ_5, i.e., the set of literals that can possibly occur before γ_5 is $s_{\gamma_5} = \{\overline{p}, q, \overline{q}\}$, where: $\overline{p}$ is part of s_{γ_5} due to the ordering constraint $(\gamma_2 \prec_\Pi \gamma_5)$; p is not included in s_{γ_5} because $\mathsf{eff}(\gamma_2)$ denies $\mathsf{eff}(\gamma_1)$

[8] We use tc to refer to the transitive closure

and $(\gamma_1 \prec_\Pi \gamma_2)$; and, q and $\overline{q}$ are part of s_{γ_5} because $\langle \gamma_3, \gamma_5 \rangle$ and $\langle \gamma_4, \gamma_5 \rangle$ are possible relations consistent with $OC(\Pi)$. Following Definition 2, we observe that $\langle \gamma_3, \gamma_4 \rangle$ belongs to the transitive closure of the relations in the plan but $\langle \gamma_4, \gamma_5 \rangle$ does not, reason why $q \in s_{\gamma_5}$. Note that Figure 8 shows a *proto-state*, s_{γ_5}, that contains an inconsistent set of literals.

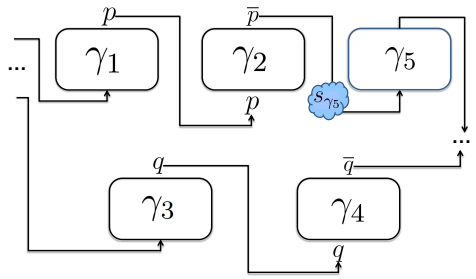

Figure 8: Example of the *proto-state* $s_{\gamma_5} = \{\overline{p}, q, \overline{q}\}$.

Back to Figure 7, the base of the attacking argument $\mathcal{A}_1$ needs to be warranted in the proto-state of the action-argument γ_2. This is graphically shown in Figure 7 by a cloud drawn under $\mathcal{A}_1$. Therefore, the tuple of the attack $\langle \mathcal{A}_0, \{\mathcal{A}_1, \mathcal{A}_2\}, \Psi \rangle$ ($\mathcal{A}_2$ is not graphically represented) is now replaced by $\langle \mathcal{A}_0, \{\mathcal{A}_1, \mathcal{A}_2\}, s_{\gamma_2} \rangle$ to ensure that all the attacking arguments are warranted in the proto-state of γ_2, the action-argument supported by $\mathcal{A}_0$. Additionally, in Figure 7(d), we have that $(\gamma_1, \ell, \mathcal{B}) \in CL(\Pi)$ and $(\mathcal{B}, z, \gamma_2) \in SL(\Pi)$. In this case, the proto-state or activation context of $\mathcal{A}_1$ would also be s_{γ_2}. The reason is that an attacking argument represents a belief of an agent against a successful execution of an action so, ultimately, the target of an attacking argument is always an action-argument step. This is also confirmed by Definition 2, where proto-states are only formed by the effects of the action-argument steps, and hence γ_1 would be indirectly supporting γ_2 in this example.

4. Cooperative Planning Protocol

In this section, we present the extension of Q-DeLP-POP to a Multi-Agent Planning (MAP) scenario, hereinafter labeled as Q-DeLP-MAP.

4.1. Multi-Agent Planning Task

Agents in Q-DeLP-MAP are entities of the problem equipped with planning and argumentation capabilities, each having its own planning and contextual information. A MAP task is composed of a finite non-empty set of n cooperative agents denoted as $\mathsf{AG} = \{\mathsf{Ag}_1 \dots \mathsf{Ag}_n\}$. Given a MAP task $\mathbb{M}$,

each agent will have a different and local view of the task, $\mathbb{M}_{Ag_i} = \langle \Psi_{\mathsf{Ag}_i}, \Delta_{\mathsf{Ag}_i}, A_{\mathsf{Ag}_i}, G \rangle$:

- $\Psi_{\mathsf{Ag}_i} \subseteq \Psi$ represents the partial view of the initial state of agent Ag_i such that $\Psi = \bigcup_{\forall \mathsf{Ag}_i \in \mathsf{AG}} \Psi_{\mathsf{Ag}_i}$ is a consistent set.
- $\Delta_{\mathsf{Ag}_i} \subseteq \Delta$ is the set of defeasible rules of agent Ag_i such that $\Delta = \bigcup_{\forall \mathsf{Ag}_i \in \mathsf{AG}} \Delta_{\mathsf{Ag}_i}$ is a set of possibly contradictory rules.
- $A_{\mathsf{Ag}_i} \subseteq A$ is the set of planning actions of agent Ag_i such that $A = \bigcup_{\forall \mathsf{Ag}_i \in A} A_{\mathsf{Ag}_i}$.
- G is the set of goals of the MAP task. Unlike the rest of elements, G is known to all the agents.

Q-DeLP-MAP is aimed at solving cooperative planning tasks where agents contribute to creating the plan with their planning actions and to ensuring the executability of the plan through their defeasible rules. Thus, agents in a MAP task jointly solve G and help discover the non-anticipated conditions that might prevent the plan from being executable.

A relevant aspect in a multi-agent system is the notion of privacy. Given two agents, Ag_i and Ag_j, they can share none, some or all of the planning actions, defeasible rules and facts of the initial state. If Ag_i and Ag_j are *functionally* distributed entities, they will have different planning capabilities. If Ag_i and Ag_j are *spatially* distributed entities, they will have different knowledge of the initial state and subsequent states of the problem. In either case, privacy is implicit in the definition of each $\mathbb{M}_{Ag_i}$. Cooperation in Q-DeLP-MAP implies that agents need each other in order to solve the goals of the MAP task or that they can rather accomplish the task better by working together [46].

If a literal l is private to agent Ag_i then l is not shared with any other agent and l can only be used by Ag_i during the plan construction and argumentative evaluation. The more private information of the agents, the less interaction between them during planning and argumentation. For this reason, in Q-DeLP-MAP scenarios, it is important that as much information as possible is labeled as public in order to promote argumentation as a coordination mechanism that enable agents to contrast their beliefs about the world towards a successful achievement of an executable plan.

Specifically, the existence of public information is more concerned with Ψ, the initial world state, and the successive planning states generated along the plan construction. However, it is commonly accepted that two agents

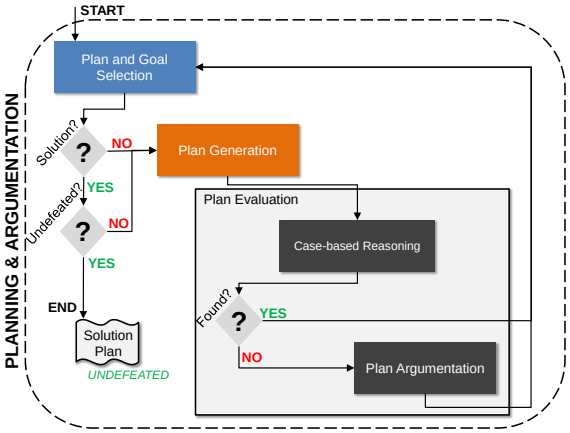

Figure 9: Main algorithm of Incremental Plan Construction in Q-DeLP-MAP.

have different planning capabilities (as commented above) and different defeasible rules. For instance, Ag_i can possess information that allows the agent to infer the traffic condition whereas Ag_j has defeasible information about the weather. Note also that, in Q-DeLP-MAP, planning data as well as the agents' beliefs denote defeasible information since, unless some privacy is involved, planning and contextual information can be refuted by the rest of the agents.

4.2. Multi-Agent Search Protocol

Figure 9 outlines the three main stages of the Q-DeLP-MAP protocol: *plan and goal selection*, *plan generation* and *plan evaluation*. Given a planning task $\mathbb{M}$, and a set of agents AG, the Q-DeLP-MAP protocol starts with an initial empty plan, $\Pi_0 = \{\alpha_\Psi \prec \alpha_G\}$, and agents progressively search through the space of plans (POP tree). At each iteration of the protocol, agents collaboratively select a node of the POP tree and expand it. The process finishes when a solution plan is found; i.e., a plan in which all step preconditions are necessarily true. The final goal of Q-DeLP-MAP is to return a solution plan with guarantees of a robust execution in the real-world.

Unlike other approaches, Q-DeLP-MAP successively interleaves planning and argumentation so as to build a plan incrementally. Once a plan Π is selected by the agents, and unless Π is labeled as a solution plan, agents select an open goal Φ of Π. Then, Π is expanded in the plan generation stage, where agents put forward and exchange refinement plans of Π that tentatively solve Φ. Subsequently, agents evaluate each plan proposal by arguing the unexpected circumstances that may occur in the context of the proposal.

4.2.1. Plan generation

A refinement plan is a plan proposal put forward by an agent as a result of expanding a selected plan Π when solving an open goal Φ. This stage follows a process similarly to a plan-space planning process that builds a POP

tree, except that each refinement or successor of Π can be now generated by a different agent and can contain arguments to support the action preconditions or arguments bases. Agents exchange their refinements between each other and learn the new steps, literals and links of the plan, which they keep in their local POP tree.

In a previous multi-agent version of DeLP-POP [31], agents engage in a turn-based dialogue permitting agents to jointly discover threats to any argument step of a refinement plan. However, in Q-DeLP-MAP, as explained in section 3.2.1), threats caused by an argument step $\mathcal{A}$ are not solved until $\mathsf{base}(\mathcal{A})$ is supported, thus enabling any other agent of AG to insert action-arguments in the plan to support $\mathsf{base}(\mathcal{A})$ and solve the threat by promotion or demotion. This is consistent with the partial-order structure of the plan and, consequently, with the local information held by the agents. Subsequently, agents will learn the new ordering constraints and will update their local POP tree accordingly. Additionally, unlike [31], Q-DeLP-MAP generates threat-free refinements of Π; that is, if an agent finds a threat in one of its refinements, the agent solves the threat before exchanging the new proposal with the rest of the agents.

The coordination protocol of Q-DeLP-MAP for agents to exchange their plan proposals is based on a *democratic leadership* where a leadership baton is scheduled among the agents following a round-robin strategy. Once the baton agent has sent its refinements to the rest of the agents and acknowledgment is received, the coordination stage is completed, and the baton is handed over to the following agent. Thus, an agent can only intervene when it holds the baton, and it is listening the rest of the time. The process is repeated until all the agents have once taken the baton role.

4.2.2. Plan evaluation

Evaluating a plan and analyzing its executability in the real world according to the agents' beliefs is performed via two protocols: *Plan Argumentation* and *Case-Based Reasoning (CBR)*. In the *Plan Argumentation* stage, agents engage in a series of argumentative dialogues aimed at evaluating the guarantee of a successful execution of the plan proposal. Specifically, argumentation occurs in the context of the agents' beliefs and arguments to defend or attack the plan proposal are built. Subsequently, the CBR will register the argumentative case in order to be able to re-use it in further evaluations.

Given a refinement or plan proposal Π_r, agents generate as many argumentative dialogues as supporting arguments and action-argument steps are present in Π. Let $\mathcal{A}^{\mathsf{Ag}_i}$ be one argument of Π_r, where Ag_i is the agent

that inserted $\mathcal{A}$ in Π_r, and let γ be the action-argument which $\mathcal{A}$ is giving support to. The argumentative dialogue to evaluate $\mathcal{A}^{\mathsf{Ag}_i}$ is encoded as a dialectical tree $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_i}}$ (see Section 2.1). Nodes of $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_i}}$ are labeled with an argument that attacks the argument in its parent node and which base is supported in the proto-state s_γ. More specifically:

1. The root node of the tree is labeled with $\mathcal{A}^{\mathsf{Ag}_i}$ such that $\mathcal{A}^{\mathsf{Ag}_i} \in AR(\Pi_r)$ or $\langle \alpha', \mathcal{A}^{\mathsf{Ag}_i} \rangle \in AA(\Pi_r)$. The argumentative process identifies the action-argument γ of Π supported by $\mathcal{A}^{\mathsf{Ag}_i}$ and creates the corresponding proto-state, s_γ.
2. A child node $\mathcal{B}^{\mathsf{Ag}_j}$ of $\mathcal{A}^{\mathsf{Ag}_i}$ represents an attacking argument against $\mathcal{A}^{\mathsf{Ag}_i}$; i.e., $\mathcal{B}^{\mathsf{Ag}_j}$ is a defeater of $\mathcal{A}^{\mathsf{Ag}_i}$. Consequently, children of $\mathcal{A}^{\mathsf{Ag}_i}$ stand for defeaters of the root argument $\mathcal{A}^{\mathsf{Ag}_i}$.
3. A child node $\mathcal{C}^{\mathsf{Ag}_z}$ of $\mathcal{B}^{\mathsf{Ag}_j}$ indicates an attack against $\mathcal{B}^{\mathsf{Ag}_j}$, so this new node is actually a supporter of the root argument $\mathcal{A}^{\mathsf{Ag}_i}$.
4. And so on.

The above process creates an argumentation line $\langle \mathcal{A}^{\mathsf{Ag}_i}, \{\mathcal{B}^{\mathsf{Ag}_j}, \mathcal{C}^{\mathsf{Ag}_z}, \ldots\}, s_\gamma \rangle$ of $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_i}}$. A dialectical tree is generated for each argument in Π_r and the leaves of the tree are undefeated arguments. Unlike the dialectical trees of a general argumentative process [15], the nodes in our dialectical tree are supported in the proto-state of the action-argument supported by the root argument. Every linear path from the root to a leaf node corresponds to one different acceptable *argumentation line*. Circular argumentation is avoided by applying that no argument can be reintroduced in the same argumentation line and that argument concordance must be guaranteed [15].

Similarly to the *Plan Generation* phase, agents can only adopt one role, namely, the baton agent or the participant agent, during the argumentative process, and the roles are iteratively swapped. The baton agent is the agent that inserted the argument in Π_r and it is the responsible for the construction of the dialectical tree. The baton agent is also allowed to put forward a self-attacking argument as well as processing the attacking/supporting arguments of the rest of agents in the dialectical tree. The *Plan Argumentation* phase of a particular argument ends when an argumentative round with no new attacking arguments is detected.

Plan argumentation is applied to every argument in all nodes of a POP tree. Let Π_r be a node which is expanded into a child node Π'_r. All the evaluated arguments in Π_r will be also comprised in Π'_r. Since the structure of a partial-order plan changes along its construction, the proto-state of some action-arguments in Π'_r may have changed too while others may have not with respect to Π_r. Thus, it seems convenient to store the results of

the *Plan Argumentation* so as to re-use them in the evaluation of refinement successors.

Case-Based Reasoning (CBR) systems allow agents to learn from their experiences [1]. Q-DeLP-MAP uses a *CBR* module with the only objective of improving the performance of the plan evaluation. A case in the argumentation database is a 4-tuple structure $\langle \mathcal{A}, \mathcal{T}_\mathcal{A}, s_\gamma, r \rangle$, where $\mathcal{A}$ is the root argument, $\mathcal{T}_\mathcal{A}$ is the dialectical of $\mathcal{A}$; s_γ is the proto-state of γ, the action-argument directly or indirectly supported by $\mathcal{A}$; and, r is the result of the evaluation of $\mathcal{A}$, that is, D (*defeated*) or U (*undefeated*). Whenever an argument is evaluated, a new 4-tuple case is inserted in the argumentation database. Likewise, before evaluating an argument of a plan refinement, we query the database to search for a similar past experience. If a similar case is found, the *Plan Argumentation* protocol is not executed and the step is directly labeled with r. Otherwise, a new argumentation phase is launched.

4.2.3. Plan and Goal Selection

An open goal from $\mathsf{goals}(\Pi)$ is selected through a heuristic function that calculates the most costly goal according to a reachability analysis based on the relaxed planning graph (RPG) of the planning task [11]. Specifically, the cost of a goal g, $cost(g)$, is estimated as the first literal level of the RPG where g appears. This is clearly an underestimation of the real cost of g since $cost(g)$ only accounts for the number of actions necessary to reach g from the initial situation with no regard of the cost of the actions preconditions or the possibly negative interactions among the actions in the plan.

As for the plan selection, Q-DeLP-MAP applies first a warranty procedure to discard the plans evaluated as *defeated* in the plan argumentation phase. Subsequently, the *undefeated* plans are estimated according to $h(\Pi) = cost(\mathsf{goals}(\Pi))$ [26]; specifically, $h(\Pi)$ is estimated as $h(\Pi) = \sum_{g \in \mathsf{goals}(\Pi)}(cost(g))$. The possibly overestimation of this additive heuristic lies in that one same action that achieves the precondition of two or more actions is counted as many times as needed actions. In contrast, this overestimation balances out the underestimation of $cost(g)$.

5. Experimental evaluation

In order to analyze the benefits and limitations of our proposal, we compare Q-DeLP-MAP with MAP-POP, a multi-agent planner without argumentation, and PS-Q-DeLP-MAP, an argumentation framework that emulates

the behavior of other approaches [8, 7][9]:

- MAP-POP is a general-purpose MAP framework suitable to cope with a wide variety of MAP domains [45]. MAP-POP is a POP-based refinement planning approach that iteratively combines planning and coordination. Agents do not have argumentation capabilities.
- PS-Q-DeLP-MAP is a modified version of Q-DeLP-MAP adapted to apply plan selection (PS) instead of applying defeasible reasoning during the search process of a plan construction. In this case, the plan generation phase is executed until completion (a solution plan that solves the problem goals is returned) and then the final plan is evaluated in an argumentative dialogue among the agents. As in [8, 7], PS-Q-DeLP-MAP draws upon a one-shot planning-argumentation approach instead of continuously interleaving planning and argumentation like in Q-DeLP-MAP. This iterative one-shot procedure is repeated until the evaluation phase returns an undefeated solution plan.

The experiments were carried out in three well-known domains used in the IPC (International Planning Competitions) benchmarks [22]:

- satellite: a space application that involves planning and scheduling a collection of observation tasks between multiple satellites, each equipped with different instruments such as infrared, spectrograph, thermograph, calibration equipment, etc.
- rovers: it is a simplification of the NASA Mars Exploration Rover problem. Multiple planetary rovers explore the environment by taking pictures, gathering samples and communicating them back to a lander.
- logistics: it involves driving trucks and flying airplanes to deliver packages between locations. Locations are either airports, reachable by planes and trucks, or places within a city, only reachable by trucks.

Since the IPC problem suites only include single-agent versions of the planning problems, we used the multi-agent version of the domains presented in [46][10]. The agentization of the satellite domain features one agent per satellite. The resulting MAP tasks are almost decoupled as each satellite can attain a subset of the task goals (even all the goals in some cases) without

[9] CAMAP is not suitable for comparison because it is an ad-hoc approach specifically designed for health-care applications

[10] Domains are also available at http://users.dsic.upv.es/grupos/grps/tools/map/fmap.html

interacting with any other agent. The agentization of the rovers domain consists in creating a planning task per rover. In this case, agents are all of the same type (rovers) and have the same set of planning actions except that only one rover is equipped with a camera to take pictures. The rovers domain generates low-level agent interaction tasks since rovers are generally capable to achieve a problem goal by themselves. The agentization of the logistics domain includes trucks and planes, thus generating a planning task per *truck* or *plane* agent with different planning capabilities. In this domain, agents need to cooperate in order to transport the packages to their target locations and tasks present several coordination points (airports) at which trucks and planes interact. Therefore, according to the agent interaction and complexity of the planning domains, satellite is the simplest domain (almost decoupled tasks), rovers offers a medium level of complexity (loosely-coupled tasks) and logistics is the most difficult domain (tightly-coupled tasks).

The planning tasks in [46] are encoded in a version of the Planning Domain Description Language (PDDL) that introduces state variables[11]. The defeasible rules[12] incorporated into the agents planning tasks of the two space applications, the satellite and rovers domains, are related to:

1. due to the existence of solar storms, one or more agents (satellites) may believe this is a probably cause for images not to be correctly taken; the existence of solar storms is also used by some agents in the rovers domains to argue about communication failures between a rover and the lander.
2. agents that have a more accurate picture of the mars surface may hold reasons to believe that a rover will not be able to move across some particular area at night. However, some rovers may be equipped with a spotlight, which would allow them to move between waypoints at night, being this information unknown to the attacking agent.
3. satellites need to calibrate their instruments to make them point at the precise direction where the image must be taken from. We defined defeasible rules that argue about the precision of the calibration models.

[11] http://ipc.informatik.uni-freiburg.de/PddlExtension

[12] Defeasible rules are encoded in the form of operators, like planning operators, through the special constructor `:def-rule` of our language (see [29]). Internally, our planner works with ground instances of operators like most planners do. Ground instances of `:def-rule` are generated using the set of literals derived from the grounding of the planning operators plus the literals of the initial situation. Subsequently, we will use the term defeasible rule to refer to any instance of a `:def-rule operator`.

4. an agent may hold reasons to believe that the usual low temperatures in the space will prevent a rover from moving to a specific waypoint; but, if the rover is equipped with a electrical heater, he would be able to refute the attack of the agent.

News like the solar storms or atmospheric conditions in the space may be likewise considered as defeasible knowledge if we assume the agent may have an outdated information. This frequently happens in space applications where rovers communicate to Earth via orbital relays and the communication from Mars to Earth has a long delay between 2.5 and 22 minutes. Thereby, depending on the time elapsed since the news was received from Earth, it can be managed as a fact or as a belief.

In the logistics domain, defeasible rules are concerned with the following knowledge held by agents:

1. weather conditions: some agents may have a more precise weather forecast or simply know that bad weather conditions may provoke delays in takeoffs or even flight cancelations.
2. environment conditions: a relatively common situation that may alter the elaboration of any plan is a call for a transport strike. Initially, this is handled as a prediction; later either the strike is confirmed or suspended. Agents can then argue about the possibility that the strike takes place and the consequences in the transport plan.

We created two types of defeasible scenarios, simple and hard, for the rovers and logistics domains and only a hard scenario for the satellite domain given its lower level of difficulty. Basically, the difference between the two scenarios relies in the number of defeasible rules that contradict each other. The overall number of defeasible rules used in each problem can be seen in Table 2 under the legend 'Def'. Particularly, the additional defeasible rules of the hard problems are designed in such a way that their heads contradict the head of some other rule of the respective simple problems. Hard scenarios introduce a higher level of disagreement among agents but this does not necessarily entail a higher number of defeated arguments since more attacks to arguments imply more rebuttals but also more supports. Defeasible rules are distributed across agents independently of the type of the agent, thus allowing agents to make inferences about any issue of the context. A detailed description on how to specify defeasible rules can be found in [29].

We tested five planning tasks for each domain, each corresponding to a particular problem of the IPC suites. Problems of the IPC are identified with names *pfile1*, *pfile2* and so on. The correspondence between our tasks and

	Problem 1 (S-pfile1, R-pfile1 and L-pfile1)				Problem 2 (S-pfile3, R-pfile3 and L-pfile3)				Problem 3 (S-pfile4, R-pfile4 and L-pfile4)				Problem 4 (S-pfile5, R-pfile5 and L-pfile5)				Problem 5 (S-pfile6, R-pfile7 and L-pfile6)			
Problems	**Ag**	**Act**	**Def**	**Go**	**Ag**	**Act**	**Def**	**Go**	**Ag**	**Act**	**Def**	**Go**	**Ag**	**Act**	**Def**	**Go**	**Ag**	**Act**	**Def**	**Go**
Satellite Hard	1S	52	52	3	2S	188	52	5	2S	253	52	8	3S	497	52	8	3S	509	52	7
Rovers Simple	1R	41	25	3	2R	52	25	3	2R	86	44	3	2R	144	44	7	3R	151	71	6
Rovers Hard	1R	41	64	3	2R	52	64	3	2R	86	123	3	2R	144	123	7	3R	151	161	6
Logistics Simple	1P 2T	56	31	4	1P 2T	80	31	6	1P 3T	133	39	7	1P 3T	156	39	8	1P 3T	174	81	9
Logistics Hard	1P 2T	56	62	4	1P 2T	80	62	6	1P 3T	133	87	7	1P 3T	156	87	8	1P 3T	174	164	9

Table 2: Configuration of the Experiments.

the IPC *pfiles* is shown in Table 2 alongside the caption of each problem. The column 'Ag' shows the number of agents (S stands for satellites, R stands for rovers, T for trucks and P for planes). 'Act' is the number of planning actions and 'Go' is the number of goals of the problem. The size of the problems increases with respect to the number of agents, actions and defeasible rules. Regarding the number of goals, only Problem 5 in the satellite and rovers domains define fewer goals than Problem 4. It is worth mentioning that the difficulty of solving a task is not only determined by the number of goals and the problem size; for instance, transporting three packages to three different destinations may be harder to solve than transporting five packages all to the same destination[13].

All the agents are executed on a single machine with a 2.83 GHz Intel Core 2 Quad CPU and 8 GB RAM (only 1 GB RAM available for the Java VM); a broker of Java Message Service[14] for the agents communication is executed in another machine also with 2.83GHz and 1GB RAM.

Sections 5.1 and 5.2 present the overall performance and quality of the solution plans, respectively. Subsection 5.3 analyzes in detail the obtained results per domain and section 5.4 shows the potential of using argumentation in planning and the level of agent contribution to each problem.

5.1. Performance analysis

The computation times obtained for the three domains are presented in Figures 10, 11(a), 11(b), 12(a) and 12(b). Some noticeable conclusions can be inferred from the figures: (a) the hardest problem in the satellite domain takes 100 seconds, 450 seconds in the rovers domain, and 800 seconds in the logistics domain, which confirms the difficulty of each domain as explained

[13]The file index is a rough indication of the problem difficulty albeit it is not always the case that a problem with a larger index is more difficult to solve.

[14]MAP-POP, Q-DeLP-MAP and PS-Q-DeLP-MAP are based on Magentix2: a platform of multi-agent systems [2].

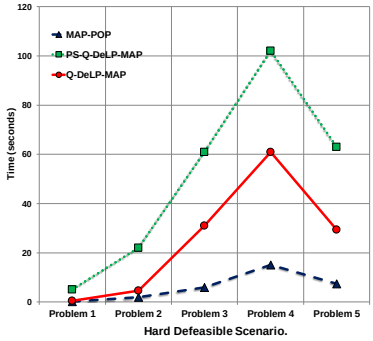

Figure 10: Average computation time of the hard scenario of the satellite domain.

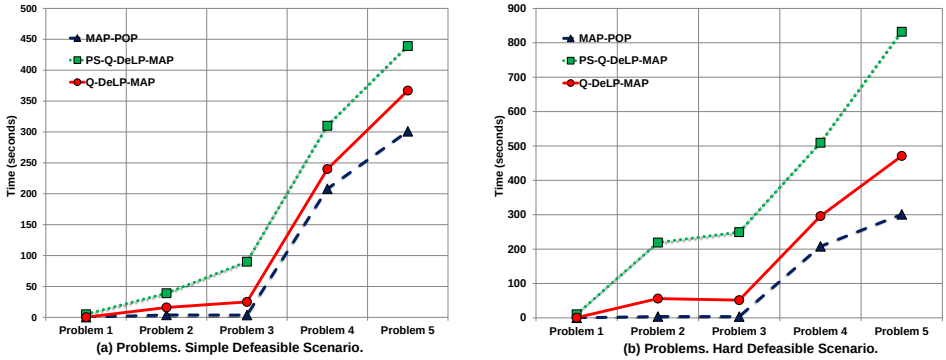

Figure 11: Average computation time of the rovers domain: (a) simple scenario (b) hard scenario

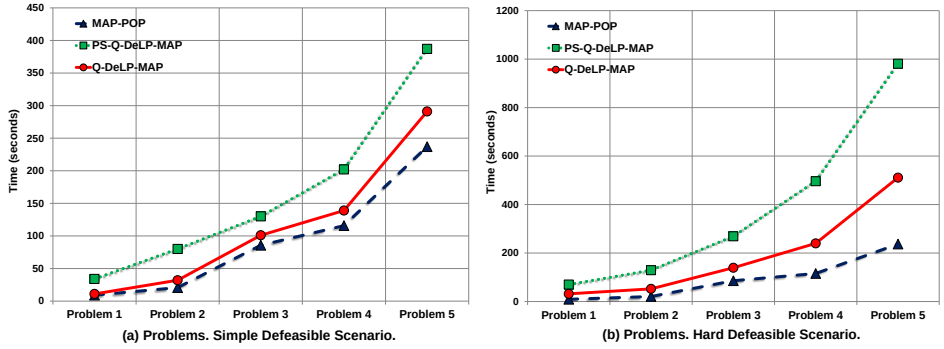

Figure 12: Average computation time of the logistics domain: (a) simple scenario (b) hard scenario)

above; (b) MAP-POP is always the most efficient approach thanks to the absence of an argumentative process; and (c) PS-Q-DeLP-MAP is always more costly than Q-DeLP-MAP.

The parameters that determine the performance results are:

1. *The size of the search space.* This is directly related to the size of the

	MAPOP					Q-DeLP-MAP					PS-Q-DeLP-MAP				
Hard Defeasible Scenario	**P1**	**P2**	**P3**	**P4**	**P5**	**P1**	**P2**	**P3**	**P4**	**P5**	**P1**	**P2**	**P3**	**P4**	**P5**
Satellite: Total number of proposed plans	1	34	71	109	33	1	63	93	322	73	4	8	15	45	9
Satellite: Total Number of Dialogues	0	0	0	0	0	35	120	240	495	163	36	88	300	675	180
Satellite: Arguments of CBR with Reuse = 0	0	0	0	0	0	23	81	179	137	83	26	57	199	445	134
Satellite: Arguments of CBR with Reuse ≥ 1	0	0	0	0	0	12	39	61	358	80	10	31	101	230	46
Rovers: Total number of proposed plans	1	54	86	450	504	1	386	423	664	606	3	9	9	18	26
Rovers: Total Number of Dialogues	0	0	0	0	0	10	341	397	2742	2632	24	90	63	360	442
Rovers: Arguments of CBR with Reuse = 0	0	0	0	0	0	10	144	158	1781	1583	20	71	55	325	412
Rovers: Arguments of CBR with Reuse ≥ 1	0	0	0	0	0	0	197	239	961	1049	4	19	8	35	32
Logistics: Total number of proposed plans	65	128	101	354	253	132	183	1366	557	1341	8	10	12	24	47
Logistics: Total Number of Dialogues	0	0	0	0	0	252	306	2562	1232	1363	144	230	396	648	1598
Logistics: Arguments of CBR with Reuse = 0	0	0	0	0	0	147	176	1355	669	819	102	192	321	387	1147
Logistics: Arguments of CBR with Reuse ≥ 1	0	0	0	0	0	105	130	1207	563	544	41	38	75	261	451

Table 3: Results of the search process for the problems in the hard defeasible scenarios.

problem (see Table 2) and affects the three approaches.

2. *Number of planning messages exchanged between agents.* It influences the performance of the three approaches and it is related to the size of the problem as well as to the complexity of the domain.
3. *Number of dialogues or dialectical trees.* It only affects Q-DeLP-MAP and PS-Q-DeLP-MAP since no argumentation is applied in MAP-POP.
4. *Number of argumentation messages exchanged between agents.* This paraemeter highly impacts Q-DeLP-MAP because the argumentative dialogues (plan evaluation) take place along the whole construction of the plan; to a lesser extent, it also affects PS-Q-DeLP-MAP when the evaluation is done at the end of the plan generation; i.e., over the solution plan.
5. *Number of times the planning process (plan generation) is executed in* PS-Q-DeLP-MAP. We used Q-DeLP-MAP as an external planner (with the argumentation stage deactivated) in order to obtain the solution plans for PS-Q-DeLP-MAP. The solution plan, which is composed of action-argument steps, is then sent to the argumentative process. If the plan turns out to be defeated, Q-DeLP-MAP is invoked again from scratch and the CBR module filters out the already discarded solution plans. This behaviour emulates exactly the usage of an external planner as in [8, 7].

Regarding the above parameters, the search results of Table 3 provide an idea of the complexity of each approach in the hard scenarios of the three domains (P1 stands for Problem1, P2 stands for Problem2 and so on):

1. The number of proposed plans in MAP-POP and Q-DeLP-MAP is the total number of nodes (partial-order plans) generated in the search

tree. Whereas Q-DeLP-MAP sends every node to the argumentative process so as to discard a potentially defeatable plan, MAP-POP only expands the nodes that result from the application of the heuristic search. Typically, Q-DeLP-MAP will generate more partial plans than MAP-POP since a node expanded by MAP-POP may be labeled as 'defeated' in Q-DeLP-MAP, in which case the node will be discarded and an alternative path will be explored. On the other hand, the number of proposed plans in PS-Q-DeLP-MAP shows the overall number of solution plans or how many times Q-DeLP-MAP has been invoked as an external planner (with the argumentative process deactivated).

2. The number of dialogues of MAP-POP is obviously 0. The difference in the number of dialectical trees between Q-DeLP-MAP and PS-Q-DeLP-MAP is due to their different behaviour: agents in PS-Q-DeLP-MAP do not exchange argumentative messages during the search process, only at the end of the plan generation.
3. The next two rows in each domain show the number of arguments that have never been reused or have been reused at least once out of the total number of evaluated arguments (dialogues).

Albeit the number of proposed plans, dialectical trees (and so the number of exchanged messages) between agents is significantly higher in Q-DeLP-MAP than in PS-Q-DeLP-MAP, PS-Q-DeLP-MAP is a more costly approach. This is because in the argumentative process of Q-DeLP-MAP, a successor node n' of a node n only introduces a new action-argument with respect to n. Thereby, if the proto-states of the plan in n' do not change with respect to n, the plan evaluation of n' will reuse all the dialogues of the parent node and only the newly added action-argument is sent for evaluation. However, the CBR module in PS-Q-DeLP-MAP reuses argumentative cases of solution plans, thus being necessary to generate a complete plan before discovering whether the solution plan is defeated or undefeated. This evidences the high computational cost of the planning machinery compared to the cost of the argumentation. As shown in Figures 10, 11 and 12, interleaving planning and argumentation is more beneficial because argumentation helps conduct the planning process and reduce the planning workload even at the cost of a higher argumentation activity.

Due to the high-level agent interaction in the logistics domain, a significantly higher number of planning messages or plan proposals appear in this domain compared to the other two domains. Additionally, the difficulty of solving the particular goal combination of each problem is a determinant factor in the number of proposed plans, reason why it is not always the case

	MAPOP					Q-DeLP-MAP / PS-Q-DeLP-MAP				
Quality Metrics of the Solution Plan	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
Satellite: Action Steps	9	11	20	15	20	8	10	18	14	18
Satellite: Duration	8	9	13	8	11	7	8	11	7	10
Rovers: Action Steps	10	12	8	24	18	8	10	7	20	17
Rovers: Duration	7	8	5	10	7	6	7	4	9	7
Logistics: Action Steps	20	25	37	31	36	18	23	33	27	34
Logistics: Duration	9	9	13	11	13	8	9	13	11	13

Table 4: Quality of the solution plans for the hard defeasible scenarios.

that the highest number of proposed plans occur in the largest problem instances. We can also observe that the correspondence between the number of proposed plans and number of dialogues differ across domains as well as in the problems of one same domain. All these aspects will be analyzed in section 5.3.

5.2. Quality of the solution plans

Table 4 shows the number of actions and the duration of the plans of the hard defeasible scenarios. Duration is measured as the number of time or execution steps of the plan. On the other hand, we count the number of actions of the plan that represent executable actions in the real world, where each action corresponds to an action-argument step in Q-DeLP-MAP and PS-Q-DeLP-MAP.

The first observation is that Q-DeLP-MAP and PS-Q-DeLP-MAP return the same plans because they use the same planning model and build dialogues similarly. The only difference is the (partial or complete) plan over which the argumentation is applied, which in turn yields different search spaces.

We can also observe in Table 4 that the number of actions in Q-DeLP-MAP and PS-Q-DeLP-MAP is always lower than in MAP-POP. The reason is that an open goal of an action-argument can be supported in PS-Q-DeLP-MAP and Q-DeLP-MAP with a supporting argument instead of an action-argument; that is, the agents' beliefs that succeed the argumentative process can be used as a support of the conditions of the planning actions. For instance, let's assume a rover situated in waypoint w_1 that plans to move to a waypoint w_2 in order to collect a soil sample. If another agent believes that a soil sample can also be found at w_1 and no other agent contradicts this information, then the agent will not need to move to w_2, thus saving one planning action. In this case, the plan will contain fewer actions because agents' beliefs are also used to support the fulfilment of an open goal.

On the other hand, the duration of the plans in Q-DeLP-MAP and PS-Q-DeLP-MAP is slightly shorter than in MAP-POP because, generally, the

fewer actions are in a plan, the fewer time steps it will comprise.

5.3. Discussion of the results per domain

Now, we jointly examine the results per domain obtained in sections 5.1 and 5.2.

Results of the satellite ***domain***. Clearly, the satellite is the simplest domain. Figure 10 shows that the computation time of problem P5 is significantly lower than the time of problem P4. This is explained because P4 is a relatively hard instance that comprises one more goal and compels two satellites to end up pointing at specific locations. Surprisingly, the number of actions of MAP-POP for P4 is 15 compared to the 20 actions of P3 and P5, an indication that determining the proper order of the operations of the two satellites is time-consuming. We can also see that in this domain, unlike the rovers and logistics domains, the number of dialogues of PS-Q-DeLP-MAP almost equals or exceeds the number of dialogues in Q-DeLP-MAP. This is due to the relatively high number of plan proposals in the satellite domain. The proposed plans are basically a combination of a plan per satellite given that the tasks are almost independent to each other. When the plan of one satellite becomes defeated, the system needs to find a new combination of a plan per agent until a successful joint global plan is obtained.

Results of the rovers ***domain***. Figures 11(a) and 11(b) show that the computation time of Problem2 and Problem3 is almost identical for the three approaches, particularly in the hard defeasible scenario. The reason is that the size of Problem2 and Problem3 is very similar, same number of agents and goals (see Table 2), as it also reveals the similar number of proposed plans of the three approaches for these two problems. We can also observe that the increase in the computation time of Problem4 and Problem5, due to the increase in the number of goals of Problem4 (7 goals) with respect to Problem3 (3 goals), is also translated into a significantly higher number of proposed plans and dialogues in all the approaches.

Results of the logistics ***domain***. Figures 12(a) and 12(b) show an almost linear increase in the computation time of the five problems. This is partly explained because, unlike the other two domains, the size of the logistics problems increase gradually (see Table 2) and so it does the size of the search space. However, the differences between the simple and hard scenarios are more pronounced in this domain and this impacts particularly negatively on PS-Q-DeLP-MAP. In logistics, agents (trucks and planes) need each other to accomplish the goal so typically a plan will contain at least a truck and a plane. The more agents involved in the plan, the more attacks

a plan will receive since defeasible rules are uniformly distributed across agents.

Interestingly, the figures for problem P3 in Q-DeLP-MAP (see Table 3) are noticeably higher than for problems P4 and P5 although the two latter contain one and two more goals, respectively. We must take into account that the goals defined in each problem are different as the destinations of the the seven, eight and nine packages to deliver in each problem, respectively, are not the same. This makes some planning configurations be harder to solve than others despite the number of goals. The high number of proposed plans in P3 stems from the fact that P3 is intrinsically a more complex problem than P4 and P5. This is also partly supported by the results shown in Table 4, where the number of actions of MAP-POP for P3 is 37 compared to the 31 actions for problem P4. Actually, P4 is a much simpler problem than P3 from the planning standpoint.

Regarding the comparison between P3 and P5, we can observe in Table 4 that MAP-POP returns a 37-action plan for P3 and a 36-action plan for P5, a consistently result with the number of proposed plans in Q-DeLP-MAP for P3 and P5 as shown in Table 3. This indicates that the difficulty of the planning process in both problems is similar. However, the number of total dialogues in P3 is noticeably higher than in P5. The reason is found in the number and type of different locations which the packages must be delivered to in the goals of each problem. Although P5 comprises the same seven packages defined in P3, destinations for these packages are not the same in both problems. Particularly, P5 defines nine packages to be delivered to four different locations (two post offices and two airports) while P3 defines seven packages to be delivered to five different locations, including three airports. Given that defeasible rules on weather conditions and potential strikes affect mainly airports, this has a stronger impact in P3, thus yielding a higher number of dialogues in Q-DeLP-MAP. In contrast, in P4 and P5, which both specify only two airports as package destinations, the number of dialogues is similar.

It is also noticeable that in P5 the number of dialogues of PS-Q-DeLP-MAP is higher than in Q-DeLP-MAP, as it occurs in the satellite domain. The justification follows the same line of explanation. The number of dialogues in PS-Q-DeLP-MAP is mainly determined by the number of proposed plans, which typically increases with the size of the problem in the number of goals, and is less sensitive to the typology of the goal combination. In this particular case, the number of exchanged messages during the argumentative process of Q-DeLP-MAP is below the number of messages needed to evaluate the 47 proposed plans in PS-Q-DeLP-MAP.

In summary, we can conclude that the performance of Q-DeLP-MAP pays off the utilization of the argumentation in solving multi-agent cooperative planning problems with respect to the other two MAP approaches.

5.4. Plan feasibility and agent contribution

Given a particular context, we say that a solution plan is feasible if it does not contain actions that would be otherwise discarded in an argumentation process. Feasibility is thus seen as a rough measure to know if a plan of MAP-POP contains actions that were labeled as failing actions in Q-DeLP-MAP and PS-Q-DeLP-MAP as a consequence of a defeated argument-action step. Thereby, feasibility is only applied to the plans returned by MAP-POP such that the higher the number of failing actions in its plans, the less robustness and guarantee of a successful execution.

Feasibility is computed as the number of actions in the plans of MAP-POP that correspond to defeated action-arguments in Q-DeLP-MAP. That is, the actions of MAP-POP that were discarded in Q-DeLP-MAP because the agents' knowledge pointed at non-anticipated conditions that would affect the execution of such actions. Table 5 shows the percentage of non-executable actions in the plans of MAP-POP for each problem in the hard defeasible scenarios of the two more complex domains, rovers and logistics. We can observe that the percentage of failing actions is significantly higher in the rovers domain. The reason is that the ratio of defeasible rules per agent is much larger in this domain and thereby more attacking arguments occur during the plan construction.

	P1	P2	P3	P4	P5
rovers	30%	25%	33%	50%	41%
logistics	15%	23%	16%	16%	22%

Table 5: Percentage of failing actions in MAP-POP in the rovers and logistics domain

Finally, we were also interested in checking the contribution level of the agents in the solution plans of Q-DeLP-MAP for the hard scenarios of rovers and logistics. This is calculated as the number of action-arguments or supporting arguments contributed by each agent to the plans. Table 6 (top) shows the contribution of the three agents (Rover 0, Rover 1 and Rover 2) to the solution plan in each problem of the rovers domain. In problem P2, one of the two rovers specified in this problem does not participate in the plan (value 0.0%). In problem P5, the three rovers collaboratively participate in the construction of the solution plan. In the logistics domain

(Table 6 (bottom)), we can observe that every agent defined in each of the problems contributes to the solution plan. It is also noticeable that, in general, the more trucks involved in the problem, the lower their level of participation since the workload can be more uniformly distributed among the total number of agents.

	P1	P2	P3	P4	P5
Rover 0	100.0%	100.0%	25.0%	29.2%	16.7%
Rover 1		0.0%	75.0%	70.8%	44.4%
Rover 2					38.9%
Truck 1	25.0%	28.0%	13.5%	32.3%	44.4%
Truck 2	50.0%	32.0%	24.3%	16.1%	8.3%
Truck 3			18.9%	16.1%	16.6%
Airplane 1	25.0%	40.0%	43.2%	35.5%	30.5%

Table 6: Percentage of agent contribution to the solution plans in the rovers domain (top) and logistics domain (bottom)

6. Conclusions and Future Work

Q-DeLP-MAP is a domain-independent argumentation-based MAP system that contributes with: (i) the extension of the DeLP-POP framework to a multi-agent setting; (ii) a novel specification of the qualification problem by means of the relationships between the argument steps and the action steps of a plan; (iii) the formalization of all the technical components of Q-DeLP-MAP; and (iv) an exhaustive experimental evaluation in three domains from the IPC and comparison with two other MAP approaches.

We conclude that using argumentation is a promising line to tackle planning problems while incorporating the agents beliefs within the reasoning process. This allows us to consider unexpected environmental conditions which cannot be modeled within the planning representation. Additionally, the extension of the model to a multi-agent system opens up many possibilities of application in real-world problems where knowledge, abilities and beliefs are distributed across several reasoning entities.

For future work, we envision several promising research directions:

(1) We are currently studying a new heuristic function that takes into account not only plan data, but also the information learned from the argumentative process. Our hypothesis is that an argumentation-based heuristic function will enable predicting the potential number of attacks of a refinement plan.

(2) In Q-DeLP-MAP, agents accept the result of the argumentation process and take it for granted. However, a wiser reasoning would consider how trustworthy the agent that puts forward the attacking/defending argument is. To this regard, we intend to design an argumentation based trust model that allows the agent to make a wiser decision in case of conflicting arguments related to a specific action, similarly to the proposal in [6] to Recommender Systems. Particularly, we will explore a model that assigns facts with a trust level according to the reliability of the agents' information source in order to estimate the trustworthiness of an attacking argument. This way, only attacks with a higher trust level would eliminate past evidences and a trust-based preference criterion will be used for selection among conflicting attacking arguments.

(3) Extending Q-DeLP-MAP to temporal planning is also a challenging objective that would involve adapting the argumentation model to temporal beliefs that change over time [27].

References

[1] A. Aamodt and E. Plaza. Case-based reasoning; foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

[2] J. M. Alberola, J. M. Such, V. Botti, A. Espinosa, and A. García-Fornes. A scalable multiagent platform for large systems. *Computer Science and Information Systems*, 10(1):51–77, 2013.

[3] L. Amgoud and S. Kaci. An argumentation framework for merging conflicting knowledge bases. *International Journal of Approximate Reasoning*, 45(2):321–340, 2007.

[4] K. Atkinson and T. J. M. Bench-Capon. Legal case-based reasoning as practical reasoning. *Artificial Intelligence and Law*, 13(1):93–131, 2005.

[5] K. Atkinson and T. J. M. Bench-Capon. Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence*, 171(10-15):855–874, 2007.

[6] P. Bedi and P. B. Vashisth. Empowering recommender systems using trust and argumentation. *Information Sciences*, 279:569–586, 2014.

[7] A. Belesiotis. *Argumentation-Based Methods for Multi-Perspective Cooperative Planning*. PhD thesis, University of Edinburgh, United Kingdom, 2012.

[8] A. Belesiotis, M. Rovatsos, and I. Rahwan. Agreeing on Plans Through Iterated Disputes. In *Proc. of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, volume 1, pages 765–772, 2010.

[9] T. J. M. Bench-Capon, K. Atkinson, and P. McBurney. Using argumentation to model agent decision making in economic experiments. *Autonomous Agents and Multi-Agent Systems*, 25(1):183–208, 2012.

[10] T. J. M. Bench-Capon and P. E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10-15):619–641, 2007.

[11] D. Bryce and S. Kambhampati. A tutorial on planning graph based reachability heuristics. *AI Magazine*, 28(1):47–83, 2007.

[12] D. Cartwright and K. Atkinson. Using computational argumentation to support e-participation. *IEEE Intelligent Systems*, 24(5):42–52, 2009.

[13] P. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.

[14] G. Ferguson and J. F. Allen. Arguing about plans: Plan representation and reasoning for mixed-initiative planning. In *Proc. of the Second International Conference on Artificial Intelligence Planning Systems (AIPS'94)*, pages 43–48, 1994.

[15] A. García and G. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(2):95–138, 2004.

[16] D. García. *Planificación y Formalización de Acciones para Agentes Inteligentes*. PhD thesis, University of Nacional del Sur, Bahía Blanca, Argentina (in Spanish), 2011.

[17] D. García, A. García, and G. Simari. Defeasible Reasoning and Partial Order Planning. In *Proc. of the 5th international conference on Foundations of information and knowledge systems (FoIKS'08)*, pages 311–328, 2008.

[18] M. L. Ginsberg and D. E. Smith. Reasoning about action II: the qualification problem. *Artificial Intelligence*, 35(3):311–342, 1988.

[19] R. Girle, D. L. Hitchcock, P. McBurney, and B. Verheij. Decision support for practical reasoning: A theoretical and computational perspective. In eds. C. Reed and T. Norman, editors, *Argumentation Machines. New Frontiers in Argument and Computation*, pages 55–84. Kluwer Academic Publishers, 2003.

[20] J. Hulstijn and L. van der Torre. Combining goal generation and planning in an argumentation framework. In *Proc. of the 10th International Workshop on Non-Monotonic Reasoning (NMR 2004)*, pages 212–218, 2004.

[21] E. Kok, J. Meyer, H. Prakken, and G. Vreeswijk. Testing the benfits of structured argumentation in multi-agent deliberation dialogues. In *Proc. of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, volume 3, pages 1411–1412, 2012.

[22] D. Long and M. Fox. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research*, 20:1–59, 2003.

[23] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *M.L. Ginsberg, ed., Readings in nonmonotonic reasoning*, pages 26–45. Morgan Kaufmann Publishers, 1987.

[24] R. Medellin-Gasque, K. Atkinson, T. J. M. Bench-Capon, and P. McBurney. Strategies for question selection in argumentative dialogues about plans. *Argument & Computation*, 4(2):151–179, 2013.

[25] A. Monteserin and A. Amandi. Argumentation-based negotiation planning for autonomous agents. *Decision Support Systems*, 51(3):532–548, 2011.

[26] X. Nguyen and S. Kambhampati. Reviving partial order planning. In *Proc of the 17th international joint conference on Artificial intelligence (IJCAI 2001)*, volume 1, pages 459–464, 2001.

[27] S. Pajares and E. Onaindia. Temporal Defeasible Argumentation in Multi-Agent Planning. In *Proc. of the 22nd International Joint Conferences on Artificial Intelligence (IJCAI 2011)*, pages 2834–2835, 2011.

[28] S. Pajares and E. Onaindia. Defeasible Argumentation for Multi-Agent Planning in Ambient Intelligence Applications. In *Proc. of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, volume 1, pages 509–516, 2012.

[29] S. Pajares Ferrando and E. Onaindia. Context-Aware Multi-Agent Planning in intelligent environments. *Information Sciences*, 227:22–42, 2013.

[30] S. Pajares Ferrando, E. Onaindia, and A. Torreño. An Architecture for Defeasible-Reasoning-Based Cooperative Distributed Planning. In *Proc. of the 19th International Conference on Cooperative Information Systems (CoopIS 2011)*, volume 7044 of *Lecture Notes in Computer Science*, pages 200–217, 2011.

[31] P. Pardo, S. Pajares, E. Onaindia, L. Godo, and P. Dellunde. Multiagent Argumentation for Cooperative Planning in DeLP-POP. In *Proc. of the 10th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, volume 3, pages 971–978, 2011.

[32] J. Penberthy and D. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proc. of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 103–114, 1992.

[33] J. Pollock. Defeasible reasoning with variable degrees of justification. *Artificial Intelligence*, 133(1):233–282, 2001.

[34] J. L. Pollock. How to reason defeasibly. *Artificial Intelligence*, 57(1):1–42, 1992.

[35] J. L. Pollock. Defeasible planning. In *Proc. of the AAAI Workshop, Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments, Technical Report WS-98-02*, pages 1–6. AAAI Press, 1998.

[36] J. L. Pollock. Rational cognition in OSCAR. In *Intelligent Agents VI, Agent Theories, Architectures*, volume 1757 of *Lecture Notes in Computer Science*, pages 71–90, 1999.

[37] I. Rahwan and L. Amgoud. An argumentation based approach for practical reasoning. In *Proc. of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pages 347–354, 2006.

[38] O. Sapena, A. Torreño, and E. Onaindia. On the construction of joint plans through argumentation schemes. In *Proc. of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, volume 3, pages 1195–1196, 2011.

[39] G. Simari and R. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 53(2-3):125–157, 1992.

[40] G. R. Simari, A. J. García, and M. Capobianco. Actions, planning and defeasible reasoning. In *Proc. of the 10th International Workshop on Non-Monotonic Reasoning (NMR 2004)*, pages 377–384, 2004.

[41] D. E. Smith and M. A. Peot. Postponing threats in partial-order planning. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI 93)*, pages 500–506, 1993.

[42] M. Thielscher. The qualification problem: A solution to the problem of anomalous models. *Artificial Intelligence*, 131(1-2):1–37, 2001.

[43] A. Toniolo, T. Norman, and K. Sycara. Argumentation schemes for collaborative planning. In *Proc. of the 14th International Conference on Agents in Principle, Agents in Practice (PRIMA 2011)*, volume 7047 of *Lecture Notes in Computer Science*, pages 323–335, 2011.

[44] A. Toniolo, T. Norman, and K. Sycara. On the benefits of argumentation schemes in deliberative dialogue. In *Proc. of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, volume 3, pages 1409–1410, 2012.

[45] A. Torreño, E. Onaindia, and O. Sapena. An approach to MultiAgent Planning with Incomplete Information. In *Proc. of the 20th European Conference on Artificiak Intelligence, Frontiers in Artificial Intelligence and Applications (ECAI 2012)*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 762–767. IOS Press, 2012.

[46] A. Torreño, E. Onaindia, and O. Sapena. FMAP: Distributed cooperative multi-agent planning. *Applied Intelligence*, 41(2):606–626, 2014.

[47] D. Walton. *Argumentation Schemes for Presumptive Reasoning.* Lawrence Erlbaum Associates, Mahwah, NJ, 1996.

[48] D. S. Weld. An introduction to least commitment planning. *AI Magazine*, 15(4):27–61, 1994.