Distributed and Multi-Agent Planning: A Survey

ALEJANDRO TORREÑO, Universitat Politècnica de València
EVA ONAINDIA, Universitat Politècnica de València
ANTONÍN KOMENDA, Czech Technical University in Prague
MICHAL ŠTOLBA, Czech Technical University in Prague

Distributed and multi-agent planning (MAP) is a relatively recent research field that combines technologies, algorithms and techniques developed by the AI planning and Multi-Agent Systems communities. While planning has been generally treated as a single-agent task, MAP generalizes this concept by considering several intelligent *agents* that plan concurrently in order to develop a course of action that satisfies the goals of the group.

This paper reviews the most relevant approaches to MAP, including the solvers that took part in the 2015 Competition of Distributed and Multi-Agent Planning, and classifies them according to the key features of the solvers, distribution and coordination.

CCS Concepts: •**Computing methodologies** → **Multi-agent planning; Cooperation and coordination;** *Planning for deterministic actions; Heuristic function construction; Multi-agent systems;* •**Security and privacy** → *Privacy-preserving protocols;*

Additional Key Words and Phrases: Distribution, planning and coordination strategies, multi-agent heuristic functions, privacy preservation

ACM Reference Format:
Alejandro Torreño, Eva Onaindia, Antonín Komenda, Michal Štolba, 2016. Distributed and multi-agent planning: a survey. *ACM Comput. Surv.* 0, 0, Article 0 (2016), 35 pages.
DOI: 0000001.0000001

1. INTRODUCTION

In general terms, the field of distributed and multi-agent planning (MAP) deals with the synthesis of plans in a multi-agent system. MAP is characterized by the distributed nature of the planning task, i.e. an environment in which planning activity is distributed across multiple agents, processes, or sites [desJardins et al. 1999], which requires computational or information distribution or a combination of both.

MAP has been extensively studied in non-deterministic settings, where outcomes are uncertain and observability of the world is limited. In this setting, MAP typically involves agents performing planning, executing their actions and receiving a local observation of the result of the actions. In decentralized control of multiple agents under uncertainty and partial observability, formal models are inspired on the use of Markov Decision Processes (MDPs) for MAP [Seuken and Zilberstein 2008].

This work is supported by the Spanish MINECO under project TIN2014-55637-C2-2-R, the Prometeo project II/2013/019 funded by the Valencian Government, and the 4-year FPI-UPV research scholarship granted to the first author by the Universitat Politècnica de València. Additionally, this research is partially supported by the Czech Science Foundation under grant 15-20433Y.
Author's addresses: A. Torreño and E. Onaindia, Universitat Politècnica de València, Camino de Vera, s/n, Valencia, 46022, Spain; A. Komenda and M. Štolba, Czech Technical University in Prague, Zikova 1903/4, 166 36, Prague, Czech Republic.

DOI: 0000001.0000001

This article surveys the existing approaches in the literature on deterministic MAP and, more particularly, in a non self-interest context. Unlike self-interested tasks, the approaches presented here are determined by the altruistic nature of the agents: agents do not seek their own benefit and private goals do not exist. Under this formalism, a MAP task defines a common goal and a set of agents to achieve such a goal. Agents will likely collaborate while accomplishing their individual goals or will work together to solve a goal that no agent is able to solve by itself [de Weerdt and Clement 2009].

The inherently distributed nature of a MAP task, the common goal of the task and the particular setting in which the planning activity takes place, determine the type of task. Thus, we distinguish between *cooperative tasks*, where two or more agents are needed to achieve a subgoal, and *collaborative tasks*, in which every subgoal can be individually solved by a single agent. Cooperative and collaborative tasks give rise to different coordination and distribution features.

A MAP task involves several challenges regarding information distribution, coordination, heuristic search or privacy preservation. The various MAP solving approaches present in the state of art can be classified according to the techniques applied to attain these challenges. In general, a MAP task solving can be interpreted as a single planning agent synthesizing a course of action for multiple executors or as a distributed activity with multiple planning agents. Moreover, the coordination of the agents' activities can be applied at different stages of the planning process, often defining the resolution scheme of the solver and affecting its efficiency when solving cooperative or collaborative tasks.

Over the last years, venues such as the Distributed and Multi-Agent Planning workshop[1] and the 2015 Competition of Distributed and Multi-Agent Planning[2] (CoDMAP) have gathered together researchers of the planning and multi-agent systems communities, making MAP a popular and rapidly developing research field. The purpose of this survey is to put in context the wide and sparse state of the art in MAP, identifying the main characteristics that define MAP tasks and solvers and establishing a taxonomy of the main works of the literature.

This survey is structured in four sections. Section 2 discusses the main features of a MAP task and its modelling. Section 3 analyzes the main aspects of MAP solvers, including distribution, coordination, heuristic search and privacy: Section 4 discusses and classifies the most relevant MAP solvers in the literature. Finally, section 5 summarizes the ongoing trends and research directions in MAP. Appendix A discloses the results of the 2015 CoDMAP, an important milestone for the MAP field, since it presents the first direct comparison of MAP solvers using a common task modelling.

2. MULTI-AGENT PLANNING TASK

We define a MAP task as a planning problem accomplished by several non self-interest agents that work jointly towards a common goal. By common goal we mean that all of the agents want the goal to be reached at the end of the task execution.

This section presents the formalization of the components of a MAP task. Next, we discuss the main aspects that characterize a MAP problem by introducing two types of MAP tasks. Finally, we present how a MAP task is modelled by using a multi-agent version of the well-known Planning Domain Description Language (*PDDL*) [Ghallab et al. 1998].

[1] http://icaps16.icaps-conference.org/dmap.html
[2] http://agents.fel.cvut.cz/codmap

2.1. Formalization of a MAP Task

We will use the MA-STRIPS formalism [Brafman and Domshlak 2008] as a basis for the formalization of a MAP task. MA-STRIPS is a minimal multi-agent extension of the classical STRIPS language [Fikes and Nilsson 1971], and it is currently the most basic and widely adopted MAP formalism.

In MA-STRIPS, a MAP task is represented through a finite number of situations or *states*. In our formalization, we use a finite set of *state variables*, denoted by $\mathcal{V}$, to describe states. A state variable $v \in \mathcal{V}$ has an associated finite domain of values, $\mathcal{D}_v$. Assigning a variable $v \in \mathcal{V}$ a value in its domain, $d \in \mathcal{D}_v$, results in a *fluent*, which we will denote as a duple of the form $\langle v = d \rangle$. A *state* S is a complete assignment to the variables in $\mathcal{V}$. The values of a state variable are mutually exclusive; consequently, if a state S includes a fluent $\langle v = d \rangle$, it does not contain any fluent of the form $\langle v = d' \rangle$, $\forall d' \in \mathcal{D}_v$, $d' \neq d$.

States change via the execution of planning *actions*. An action in MA-STRIPS is defined as follows:

Definition 2.1. A **planning action** is a tuple $\alpha = \langle pre(\alpha), eff(\alpha), cost(\alpha) \rangle$, where $pre(\alpha)$ and $eff(\alpha)$ are partial assignments to the variables of $\mathcal{V}$ that describe the preconditions and effects of the action α, respectively, and $cost(\alpha)$ denotes the cost of α.

An action α can be executed in a state S if and only if all its preconditions hold in S; that is, $\forall p \in pre(\alpha)$, $p \in S$. The execution of α in S generates a state S' such that $eff(\alpha) \subseteq S'$: given a fluent $\langle v = d \rangle \in S$ and an action α with an effect $\langle v = d' \rangle$, the application of α results in a state $S' = S \cup \langle v = d' \rangle \setminus \langle v = d \rangle$.

Definition 2.2. A **MAP task** is defined as a 5-tuple $\mathcal{T} = \langle \mathcal{AG}, \mathcal{V}, \{\mathcal{A}\}_{i=1}^{n}, \mathcal{I}, \mathcal{G} \rangle$ with the following components:

— $\mathcal{AG}$ is a finite set of n planning entities or agents.
— $\mathcal{V}$ is a finite set of state variables.
— $\mathcal{A}^i$ represents the planning actions executable by an agent $i \in \mathcal{AG}$.
— $\mathcal{I}$ is a complete assignment to $\mathcal{V}$ that denotes the *initial state* of $\mathcal{T}$.
— $\mathcal{G}$ is a partial assignment to $\mathcal{V}$ that denotes the common goal of $\mathcal{T}$.

A *solution plan* for a MAP task $\mathcal{T}$ is a partially-ordered plan $\Pi = \{\Delta, \mathcal{CL}, \mathcal{OR}\}$, where $\Delta = \{\alpha_1^i, \alpha_2^j, \ldots\} \subseteq \mathcal{A}$ is the set of actions of the plan, which can be contributed by several agents in $\mathcal{AG}$. $\mathcal{CL}$ and $\mathcal{OR}$ are two sets of *causal links* and *partial orderings*, respectively, which establish dependencies among the actions in Δ. The application of the actions of Π in the initial state $\mathcal{I}$ leads to a state S_G that satisfies the task goals, $\mathcal{G} \subseteq S_G$. A solution is cost-optimal if it has minimal total cost over all the possible solutions.

The information of a MAP task is distributed among the planning agents. Each agent $i \in \mathcal{AG}$ has a *local view* or local task of $\mathcal{T}$, $\mathcal{T}^i$, which is formally defined as follows:

Definition 2.3. The **local view** of a task $\mathcal{T}$ by an agent $i \in \mathcal{AG}$ is defined as $\mathcal{T}^i = \langle \mathcal{V}^i, \mathcal{A}^i, \mathcal{I}^i, \mathcal{G} \rangle$, which includes the following elements:

— $\mathcal{V}^i$ denotes the state variables that agent i knows.
— $\mathcal{A}^i \subseteq \mathcal{A}$ is the set of planning actions executable by i.
— $\mathcal{I}^i$ is the set of fluents that are initially known to agent i.
— $\mathcal{G}$ denotes the common goal of the task $\mathcal{T}$. An agent i knows all the items of $\mathcal{G}$, and it will contribute to their achievement either directly (achieving a goal $g \in \mathcal{G}$) or indirectly (achieving subgoals whose effects help other agents achieve g).

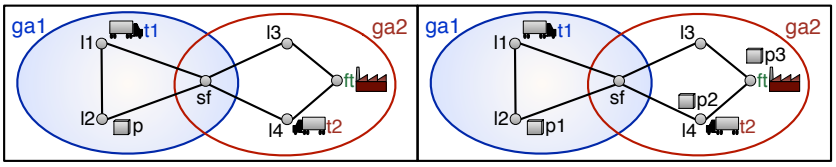

Fig. 1. Cooperative (left) and collaborative (right) MAP task examples

Note that we do not state $\mathcal{G}^i$ because $\mathcal{G}$ is not necessarily assigned to some particular agents (see next section for more details). Additionally, $\mathcal{T}^i$ determines the values of the domain $\mathcal{D}_v$ of a variable $v \in \mathcal{V}$ known to agent i, denoted as $\mathcal{D}_v^i \subseteq \mathcal{D}_v$.

2.2. Characterization of a MAP Task

In order to practically illustrate the features of a MAP task, let us introduce two different application examples based on a logistics domain [Torreño et al. 2014b].

Example 2.4. **(task $\mathcal{T}_1$)** Consider the transportation task in Figure 1 (left), which includes three different agents. There are two transport agencies, $ta1$ and $ta2$, each of them having a truck, $t1$ and $t2$, respectively. The two agencies work in two different geographical areas, $ga1$ and $ga2$, respectively. The third agent is a factory, ft, located in the area $ga2$.

To manufacture products, factory ft requires a package of raw materials, p, which is collected from area $ga1$. In this task, agents are specialized: $ta1$ and $ta2$ have the same planning capabilities, but they act in different geographical areas; i.e., they are spatially distributed agents. Moreover, the factory agent ft is functionally different from $ta1$ and $ta2$.

The goal of task $\mathcal{T}_1$ is for ft to manufacture a final product fp. In order to solve $\mathcal{T}_1$, $ta1$ will send its truck $t1$ to load the package of raw materials p, initially located in $l2$, and then $t1$ will transport p to a storage facility, sf, that is located at the intersection of both geographical areas. Then, $ta2$ will complete the delivery by using its truck $t2$ to transport p from sf to ft, which will in turn manufacture the final product fp. Therefore, this task involves three specialized agents that are spatially and functionally distributed and must cooperate to accomplish the common goal of having a final product fp.

Example 2.5. **(task $\mathcal{T}_2$)** The task in Figure 1 (right), is based on the same network of cities and includes the same three agents of task $\mathcal{T}_1$; that is, the two transport agencies $ta1$ and $ta2$ and the factory ft.

Task $\mathcal{T}_2$ features three different packages, $p1$, $p2$ and $p3$. The task goals are as follows: packages $p1$ and $p2$ must be delivered to the storage facility sf by the transport agencies. In turn, the factory ft must produce a final product fp from the package of raw materials $p3$ already located in ft.

Unlike Example 2.4, cooperation among agents is not required to solve task $\mathcal{T}_2$, since agents $ta1$ and $ta2$ can individually transport the packages $p1$ and $p2$, respectively, to the storage facility sf. Moreover, package $p3$ is initially reachable by ft, and thereby the factory agent ft can manufacture the final product fp by itself.

The MAP tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ outline two examples that present distinct characteristics and interactions among the agents. A MAP task can be classified as either *cooperative* or *collaborative* depending on the level of cooperation needed for its resolution. Given a MAP task $\mathcal{T}$:

— $\mathcal{T}$ is a *cooperative task* if $\mathcal{G}$ includes at least a *cooperative goal*. A goal $g \in \mathcal{G}$ is said to be cooperative if, given the task $\mathcal{T}_g = \langle \mathcal{AG}, \mathcal{V}, \mathcal{A}, \mathcal{I}, g \rangle$ that has g as a single goal, and for any agent $i \in \mathcal{AG}$, there is no solution plan $\Pi_g = \{\Delta, \mathcal{CL}, \mathcal{OR}\}$ in which $\forall \alpha \in \Delta, \alpha \in \mathcal{A}^i$. That is, a cooperative goal is an objective that cannot be achieved by a single agent because its resolution implies the cooperation of two or more specialized agents. Thus, cooperation among agents is a requirement in order to attain a cooperative task.
In task $\mathcal{T}_1$ (see Example 2.4), agent ft is not able to manufacture the final product fp unless agent $ta2$ delivers the package of raw materials p to ft; additionally, agent $ta2$ requires $ta1$ to deliver first p in sf. All in all, cooperation among the three participating agents is necessary to solve task $\mathcal{T}_1$.

— $\mathcal{T}$ is a *collaborative task* if, for every $\mathcal{T}_g = \langle \mathcal{AG}, \mathcal{V}, \mathcal{A}, \mathcal{I}, g \rangle$ that has a single goal $g \in \mathcal{G}$, there exists at least one solution plan $\Pi_g = \{\Delta, \mathcal{CL}, \mathcal{OR}\}$ where all the actions $\alpha \in \Delta$ are contributed by a single agent $i \in AG$; i.e., $\forall \alpha \in \Delta, \alpha \in \mathcal{A}^i$.
The above definition states that there must be at least one solution plan for task $\mathcal{T}$ in which every $g \in \mathcal{G}$ is achieved by a single agent in isolation. Obviously, other solutions where two or more agents collaborate in the achievement of a goal g may also exist, as an indication that $\mathcal{T}_g$ is likely to be solved better with collaboration.
Task $\mathcal{T}_2$ (see Example 2.5) includes three different goals that are solvable by a single agent: agents $ta1$ and $ta2$ can individually deliver packages $p1$ and $p2$ to sf, whereas the factory ft does not require any assistance to manufacture the final product fp from $p3$, since $p3$ is already in ft in the initial situation. Thus, agents are not required to cooperate because each of them can autonomously solve their goals. However, the joint realization of the agents' plans may reveal situations in which agents leverage the effects achieved by others or collaborate to avoid conflicts in the synchronization of their plans.

Table I. Task view $\mathcal{T}^i$ for each agent i in example tasks $\mathcal{T}_1$ and $\mathcal{T}_2$

Task	$\mathcal{T}_1$ (Example 2.4)			$\mathcal{T}_2$ (Example 2.5)		
Agent	$ta1$	$ta2$	ft	$ta1$	$ta2$	ft
$\mathcal{V}^i$	$pos(t1)$	$pos(t2)$	$pending(fp)$	$pos(t1)$	$pos(t2)$	$pending(fp)$
	$at(p)$ $manufactured(fp)$			$at(p1)$, $at(p2)$, $at(p3)$ $manufactured(fp)$		
$\mathcal{A}^i$	drive, load, unload		manufacture	drive, load, unload		manufacture
$\mathcal{I}^i$	$pos(t1) = l1$ $at(p) = l2$	$pos(t2) = l4$	$pending(fp) = true$	$pos(t1) = l1$ $at(p1) = l2$	$pos(t2) = l4$ $at(p2) = l4$ $at(p3) = ft$	$pending(fp) = true$ $at(p3) = ft$
	$manufactured(fp) = false$			$manufactured(fp) = false$		
$\mathcal{G}$	$manufactured(fp) = true$			$at(p1) = sf$ $at(p2) = sf$ $manufactured(fp) = true$		

Tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ emphasize most of the key elements of a MAP task. The spatial and/or functional distribution of planning agents gives rise to *specialized agents* that have different capabilities and knowledge of the task. Information of the MAP tasks is distributed among the specialized agents as summarized in Table I.

The state variables of these two tasks are distributed among the planning agents, as described in Table I. $pos(t1)$ is a variable of agent $ta1$, such that $\mathcal{D}_{pos(t1)} = \{l1, l2, sf\}$, which models the position of truck $t1$. $ta2$ has the variable $pos(t2)$, where $\mathcal{D}_{pos(t2)} = \{sf, l3, l4, ft\}$, describing the location of truck $t2$. Finally, the variable $pending(fp)$,

$\mathcal{D}_{pending(fp)} = \{true, false\}$, belongs to agent ft and denotes whether the manufacturing of the final product fp, the goal of task $\mathcal{T}_1$ and one of the three goals of task $\mathcal{T}_2$, is pending or not.

Variables related to the location of the products, $at(p)$ in task $\mathcal{T}_1$ and $at(p1)$, $at(p2)$ and $at(p3)$ in $\mathcal{T}_2$, as well as $manufactured(fp)$, $\mathcal{D}_{manufactured(fp)} = \{true, false\}$, which describes whether the final product fp is already manufactured or not, are known to the three agents $ta1$, $ta2$ and ft. Since agents ignore the configuration of the working area of the other agents, the domain of the variable $at(p)$ in $\mathcal{T}_1$, $\mathcal{D}_{at(p)} = \{l1, l2, t1, sf, l3, l4, t2, ft\}$, is distributed among the agents as follows: $\mathcal{D}^{ta1}_{at(p)} = \{l1, l2, t1, sf\}$, $\mathcal{D}^{ta2}_{at(p)} = \{sf, l3, l4, t2, ft\}$ and $\mathcal{D}^{ft}_{at(p)} = \{ft\}$. The domain of variables $at(p1)$, $at(p2)$ and $at(p3)$ is also distributed among agents in task $\mathcal{T}_2$.

The planning agents in tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ are specialized, and thereby, they can apply different actions (see Table I): the capabilities of the transport agencies $ta1$ and $ta2$ involve *loading* and *unloading* packages, as well as *driving* the trucks between locations, while the factory ft is limited to *manufacturing* final products. Additionally, transport agents are spatially distributed: agent $ta1$ only drives its truck $t1$ within the geographical area $ga1$, while $ta2$ is limited to area $ga2$.

The distribution of the task information in MAP stresses the issue of *privacy*, which is one of the basic aspects that must be considered in multi-agent applications [Serrano et al. 2013]. Since the three parties involved in tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ are specialized in different functional sectors of the task, most of the information managed by factory ft is not relevant for the transport agencies and vice-versa. The same happens with the transport agencies $ta1$ and $ta2$, which are geographically distributed.

Moreover, agents may not be willing to disclose sensitive information of their internal procedures with the others. For instance, $ta1$ and $ta2$ are cooperating in these particular delivery tasks, but they might be potential competitors since they work in the same business sector. For these reasons, agents do not unveil each other the internal configuration of their working areas, which is described through the state variables $pos(t1)$, $pos(t2)$ and $at(p)$ in task $\mathcal{T}_1$. While $pos(t1)$ and $pos(t2)$ are private to agents $ta1$ and $ta2$, respectively, the domain of $at(p)$, $\mathcal{D}_{at(p)}$, is distributed among the three participants to enforce privacy. This can be observed in Table I, where $\mathcal{I}^{ft}$ does not include either the location of the trucks or the packages in any of the two tasks. We must note that agent ft does know about the existence of the packages and the trucks but it ignores their initial location because the geographical areas of $ta1$ and $ta2$ are unknown to ft.

All in all, agents in MAP want to minimize the information they share with each other, either for strategic reasons or simply because their knowledge is not relevant for other participating agents to solve the MAP task.

2.3. Modelling of a MAP Task

Modelling a MAP task involves defining several elements that are not present in single-agent planning tasks. Widely-adopted single-agent planning specification languages, such as *STRIPS* [Fikes and Nilsson 1971] or the Planning Domain Definition Language (*PDDL*) [Ghallab et al. 1998], lack the required machinery to specify a MAP task.

One way of modelling a MAP task is via an *unfactored* specification, where the planner receives a single and complete description of MAP task $\mathcal{T}$. However, some approaches to MAP require local *factored* descriptions of one planning task instance per agent, meaning that each agent $i \in \mathcal{AG}$ receives as input a description of $\mathcal{T}^i$, its local view of the task. Additionally, the modelling of a MAP task may include information

regarding agents' privacy; that is, the information that the agent can and cannot share with other planning agents.

In the literature, *MA-STRIPS* [Brafman and Domshlak 2008] is identified as one of the earliest and most widely adopted MAP languages. *MA-STRIPS* was designed as a minimalistic multi-agent extension to *STRIPS* [Fikes and Nilsson 1971] to define various planning agents along with the actions executable by each agent.

Authors of MAP-POP and FMAP [Torreño et al. 2014a; 2015] introduce a MAP definition language that models the world states through state variables instead of predicates. The FMAP language is built upon the most recent revision of *PDDL*, *PDDL3.1* [Kovacs 2011], and allows for factored task descriptions. The domain description has the structure of a regular *PDDL3.1* domain, and the problem description is extended with an additional `:shared-data` construct, which describes the information of an agent that is shareable with the others.

The recent *MA-PDDL*[3], developed in the context of the 2015 CoDMAP, stands out as the first attempt to create a *de facto* standard specification language for MAP tasks. Similarly to the language used by MAP-POP and FMAP, *MA-PDDL* extends *PDDL3.1* to a multi-agent context, and allows for the definition of factored (`:factored-privacy` requirement) and unfactored (`:unfactored-privacy` requirement) task representations.

In order to model the transportation tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ (see Examples 2.4 and 2.5, and Figure 1 in Section 2.2) with *MA-PDDL*, we will use the factored specification, by which the planning task $\mathcal{T}^i$ of an agent i is encoded with two independent files containing the description of its domain ($\mathcal{V}^i$, $\mathcal{A}^i$) and its problem ($\mathcal{I}^i$, $\mathcal{G}$). For the sake of simplicity, we only display fragments of the task views of agents $ta1$ and ft, since $ta1$ and $ta2$ are functionally equivalent. It is worth noting that all the actions in tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ have unitary costs; i.e., $\forall \alpha \in \mathcal{A}$, $cost(\alpha) = 1$.

First, we show the encoding of the *domain* block for `agency` and `factory` agents, that appears in both $\mathcal{T}_1$ and $\mathcal{T}_2$. Then, we describe the *problem* block encoding for task $\mathcal{T}_2$.

The domain description of agents of type transport `agency`, like $ta1$ and $ta2$, is defined in Listing 1.

```
(define (domain agency)
  (:requirements :factored-privacy :typing :equality :fluents)
  (:types area location package agency - object
          truck place - location)
  (:predicates
    (manufactured ?p - product)
  )
  (:functions
    (at ?p - package) - location
    (:private
      (link ?p1 - place) - place      (owner ?a - agency) - truck
      (in-area ?p - place) - area     (pos ?t - truck ?l) - location
    )
  )
  (:action drive
    :parameters   (?a - agency ?t - truck ?p1 - place ?p2 - place)
    :precondition (and (= (in-area ?p1) ?a) (= (in-area ?p2) ?a)
                  (= (owner ?a) ?t) (= (pos ?t) ?p1) (= (link ?p1) ?p2))
    :effect       (assign (pos ?t) ?p2)
  )
[...]
)
```

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

Listing 1. Excerpt of the domain file for agency agents in tasks $\mathcal{T}_1$ and $\mathcal{T}_2$

The domain encoding of transport agencies $ta1$ and $ta2$ starts with the definition of the type hierarchy, which includes the agency type to define the transport agents. The domain description includes only one predicate, manufactured, which is used to know whether the task goal of manufacturing a product is fulfilled or not. Despite the fact that only the factory agent ft has the ability to manufacture products, all the agents know the predicate manufactured so that the transport agencies can be informed when the task goal is achieved.

Within the function names, the function at models the position of the packages and is included in the $\mathcal{T}^i$ of the three agents. The rest of functions in Listing 1 are private to $ta1$ and $ta2$, which prevents the agencies from disclosing internal information, such as the places that compose the agent's working area (in-area function), the connections among these places (link function), the agency's trucks (owner function) and the trucks locations (pos function).

Since agents in tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ are specialized, they have different planning operators. Agents $ta1$ and $ta2$ have three operators: load, unload and drive. Listing 1 shows the encoding of the drive operator, to drive a truck between two different places. The movements of the trucks of a transport agency are limited to their working area, either ga1 or ga2, through the in-area preconditions. An agency can only drive a truck if it is the owner of such resource.

Listing 2 showcases the domain file for factory agents, such as agent ft in tasks $\mathcal{T}_1$ and $\mathcal{T}_2$. The type hierarchy includes the type factory. This is a subtype of place because a factory is also interpreted as a place reachable by a truck.

```
(define (domain factory)
  (:requirements :factored-privacy :typing :equality :fluents)
  (:types location package product - object
          truck place - location
          factory - place)
  (:predicates
    (manufactured ?p - product)
    (:private
      (pending ?p - product)
    )
  )
  (:functions
    (:private
      (at ?p - package) - location
    )
  )
  (:action manufacture
    :parameters   (?f - factory ?p - package ?fp - product)
    :precondition (and (= (at ?p) ?f) (pending ?fp))
    :effect       (and (not (pending ?fp)) (manufactured ?fp))
  )
)
```

Listing 2. Domain file for factory agents in tasks $\mathcal{T}_1$ and $\mathcal{T}_2$

The predicates of the factory agent ft include both the manufactured predicate (this predicate is defined in the domain files of the three agents), and the private pending predicate to keep track of the factory's pending orders. The at function is used to notify ft about the arrival of new packages. This function is defined as private in this domain because the factory does not need to inform the rest of agents about the position of the packages that are delivered to ft. Finally, the factory ft has only one operator to manufacture pending products. This operator requires a package of raw materials to be delivered to the factory for the action to be executed.

Regarding the *problem* description, in a factored specification we create three problem files, one per agent, each including $\mathcal{I}^i$, the initial state of each agent $i \in \mathcal{AG}$, and the task goals $\mathcal{G}$. Listings 3 and 4 depict the problem description of task $\mathcal{T}_2$ for agents $ta1$ and ft, respectively.

```
(define (problem ta1)
  (:domain agency)
  (:objects
    ta1 ta2 - agency        ft - factory
    ga1 - area              t1 - truck
    p1 p2 p3 - package      l1 l2 sf - place
  )
  (:init
    (= (pos t1) l1)(= (owner t1) ta1)(= (at p2) l4)(= (at p3) ft)
    (= (link l1) l2)(= (link l2) l1)(= (link l1) sf)
    (= (link sf) l1)(= (link l2) sf)(= (link sf) l2)
    (= (in-area l1) ga1)(= (in-area l2) ga1)(= (in-area sf) ga1)
  )
  (:goal
    (and (manufactured fp)(= (at p1) sf)(= (at p2) sf))
  )
)
```

Listing 3. Problem file of agent $ta1$ in task $\mathcal{T}_2$

The problem block of $ta1$, displayed on Listing 3, defines the objects known to such agent. More precisely, this transport `agency` knows its `truck t1`, along with the `places` within its working `area`, `ga1`, and the `packages p1`, `p2` and `p3`.

Agent $ta1$ knows the initial state of its working `area ga1` (see `:init` section of Listing 3); i.e., the position of of its `truck t1` and the `packages p2` and `p3`, which are initially located in `ga1`. The location of `package p1` is unknown to $ta1$ because `p1` is initially placed in `area ga2`. Additionally, $ta1$ is aware of the `links` between the `places` within `ga1`: `l1`, `l2` and `sf`.

The `:goal` section is common to the three participating agents and includes the three common goals of task $\mathcal{T}_2$, namely a goal indicating that the `product fp` must be `manufactured`, and two goals that describe the desired final location of `packages p1` and `packages p2`; that is, the storage facility `sf`.

```
(define (problem ft)
  (:domain factory)
  (:objects
    ta1 ta2 - agency        ft - factory
    p1 p2 p3 - package      fp - product
  )
  (:init
    (= (at p3) ft)(pending fp)
  )
  (:goal
    (and (manufactured fp)(= (at p1) sf)(= (at p2) sf))
  )
)
```

Listing 4. Problem file of agent ft in task $\mathcal{T}_2$

Finally, Listing 4 features the problem description for the `factory` agent ft. First, the objects known to ft, that is, the `packages p1`, `p2` and `p3` and the `product fp`. The initial state of agent ft defines the initial location of `package p3` and the current status of the final `product fp` (`pending` to be `manufactured`). As stated above, the problem file of agent ft includes the same goal definition as for agents $ta1$ and $ta2$.

This modelling example shows the flexibility of *MA-PDDL* for encoding the specific requirements of a MAP task, such as the agents' distributed information via factored input and the private aspects of the task, including predicates, functions and operators.

Thus, the unfactored and local factored input provided by *MA-PDDL* along with the rest of functionalities, make this language be fairly expressive so as to allow different ways of specifying a MAP task.

3. PRINCIPAL ASPECTS OF A MAP SOLVER

MAP tasks involve various additional challenges than the more compact single-agent planning task formulation, such as information distribution, specialized agents, coordination or privacy. The different MAP solving techniques in the literature can be classified according to the mechanisms applied to address these challenges. Here, we highlight the four most relevant aspects according to which MAP solvers can be categorized:

— **Distribution of a MAP solver**: In general, MAP is conceptually concerned with planning *for* several execution agents by means of a single planning entity, or planning *by* multiple agents. In general, approaches that consider a single planning entity leverage the distributed structure of a MAP task to improve the efficiency of the solving process carried out by the planner. In contrast, planning by multiple agents involves several entities that jointly solve the MAP task, which implies the design of robust communication protocols, privacy-preserving planning algorithms, and distributed heuristic functions.
From a development perspective, MAP solvers use either a centralized or monolithic implementation that solves the MAP task in a self-contained fashion, or distribute the planning activity among several software agents, focusing on the coordination and synchronization of several decentralized entities [Štolba et al. 2016b].
— **Coordination strategies and resolution schemes**: A MAP task can be viewed as the problem of coordinating agents in a shared environment where information is distributed [Torreño et al. 2014b]. Coordination is applied at different points of the MAP solving process, resulting in a wide variety of resolution schemes.
We will categorize as *unthreaded* planning and coordination those solvers that apply planning and coordination as two separate and clearly identified stages. Other frameworks, however, continuously combine planning and coordination by *interleaving* both stages. In this scheme, agents jointly build solution plans in a coordinated fashion.
— **Heuristic search**: Most approaches to MAP use some kind of heuristic search. In a distributed MAP context, heuristic functions are generally applied by each agent locally, which diminishes the accuracy of the estimates due to the limited view that agents have over the MAP task. One of the current challenges of MAP focuses on the development of *global heuristics* that match the accuracy of single-agent heuristic functions.
— **Privacy preservation**: Privacy is one of the main motivations to adopt a MAP approach. Privacy means coordinating agents without making sensitive information publicly available. Whereas this aspect was initially neglected in MAP [van der Krogt 2009], the most recent approaches tackle this issue through the development of robust privacy-preserving algorithms.

The following subsections provide an in-depth analysis of these aspects, which characterize and determine the performance of the existing MAP solvers when solving MAP tasks of a given typology.

3.1. Distribution of a MAP solver

The most basic criterion to classify the MAP solvers in the literature is the *distribution* of the planning process, which can be viewed from two different perspectives. From a

conceptual viewpoint, solvers are classified according to the number of planning agents they use ($|\mathcal{AG}|$). From a development perspective, solvers divide into *centralized* and *distributed*.

Table II. Conceptual problem solving schemes according to the number of planning and execution agents

		Planning agents $\|\mathcal{AG}\|$	
		1	n
Execution agents	1	Single-agent planning FD [Helmert 2006]	Factored planning ADP [Crosby et al. 2013]
	n	Planning *for* multiple agents TFPOP [Kvarnström 2011]	Planning *by* multiple agents FMAP [Torreño et al. 2015]

3.1.1. Conceptual Problem Solving. A MAP task $\mathcal{T}$ includes the definition of a set of planning agents $\mathcal{AG}$, as formalized in Definition 2.2. In a MAP context, it is typically assumed that each planning agent in $\mathcal{AG}$ has a corresponding executor that performs its assigned actions in the solution plan Π. However, many solvers in the literature alter this balance by modifying the cardinality of $|\mathcal{AG}|$.

Table II summarizes the different problem solving schemes according to the relation between the number of planning agents and execution agents. Single-agent planning is the simplest mapping: in this scheme, a task is solved by a single planning agent, i.e., $|\mathcal{AG}| = 1$, and the obtained solution plan will be later executed by a single executor. In this case, there exists a one-to-one correspondence between planning and execution agents.

As shown in Table II, MAP solvers like ADP [Crosby et al. 2013], Distoplan [Fabre et al. 2010] and A# [Jezequel and Fabre 2012], follow a *factored planning* scheme. This technique decomposes a single-agent task into a set components or factors (agents) separately, giving rise to a MAP task with $|\mathcal{AG}| > 1$. Then, factored methods to compute local plans are applied, and finally, the local plans are pieced together into a valid global plan [Brafman and Domshlak 2006]. Factored planning exploits locality by computing solutions locally and propagating limited information between components.

The second row of Table II outlines the classification of MAP approaches aimed at building a plan that will be executed by several agents. Thus, some solvers in the literature conceive MAP as planning *for* a set of execution agents; i.e., a single planner ($|\mathcal{AG}| = 1$) calculating a plan for multiple executors. Other approaches regard MAP as planning *by* multiple agents ($|\mathcal{AG}| > 1$), generally assuming a one-to-one mapping between planners and executors. This distinction gives rise to a broad range of approaches to MAP.

Planning for *multiple agents.* Several works in the literature apply a *single-planner* approach to MAP, defining only one planning entity that synthesizes a global plan *for* a set of execution agents. Under this one-to-many correspondence scheme, the single planning agent has complete knowledge of the MAP task $\mathcal{T}$.

Under this approach, a common way of distributing the actions among the executors is through the introduction of constraints. The single planning agent defined in TFPOP [Kvarnström 2011] applies a custom forward POP search that maintains a sequence of actions per executor within the partial-order plan. This allows each execution agent to perform its actions sequentially while executing in parallel with the rest of agents.

The main limitation of this MAP scheme is its lack of privacy, since the single planning entity must have complete knowledge of the MAP task $\mathcal{T}$. This is rather unrealistic if the agents involved in the task have sensitive private information that they are

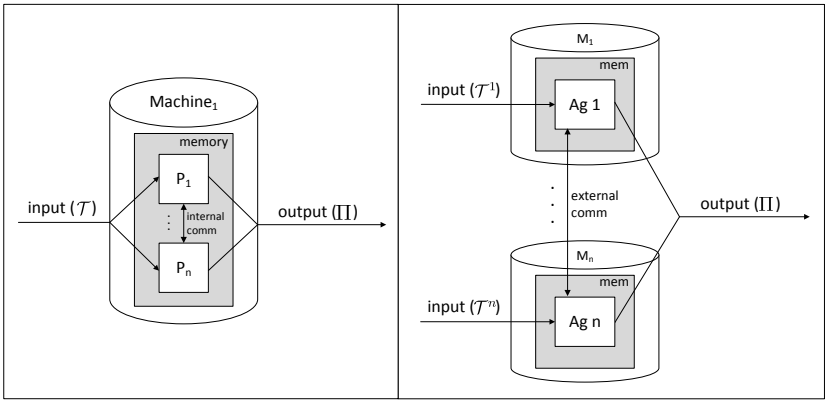

Fig. 2. Centralized (monolithic) vs. distributed (agent-based) implementation

not willing to disclose [Sapena et al. 2008]. For instance, transport agencies in Examples 2.4 and 2.5 wish to conceal the information regarding their internal processes and business strategies; therefore, a single-planner approach is not an acceptable solution in this particular task.

Planning by *multiple agents*. A wide range of methods in the literature conceive MAP as a task simultaneously performed *by* a group of independent planning entities in which the planning capabilities are likely to be distributed among agents; that is, each agent $i \in \mathcal{AG}$ has an associated task view $\mathcal{T}^i$.

In this approach, the focus is on the coordination of the activities of the various planning agents. Unlike single-planner approaches, the planning decentralization inherent to this scheme makes it possible to effectively preserve the agents' privacy.

Most MAP solvers that follow this scheme maintain a one-to-one correspondence between planners and executors. However, there exist some exceptions in the literature such as MARC [Sreedharan et al. 2015]. This solver rearranges the n planning agents in $\mathcal{AG}$ into m transformer agents ($m < n$), thus resulting in a new set $\mathcal{AG}_{MARC}$ such that $|\mathcal{AG}_{MARC}| < |\mathcal{AG}|$. Therefore, MARC breaks the one-to-one correspondence between planning and execution agents in this planning-by-multiple-agents scheme.

3.1.2. Development Perspective. From a development perspective, MAP solvers can be classified as either *centralized* or *distributed*. Centralized MAP solvers draw upon a monolithic implementation that synthesizes a global solution plan for the MAP task by means of a self-contained execution. On the contrary, distributed MAP methods are implemented as multi-agent systems in which the execution is distributed among a set of actual *software agents*, thus putting the focus on the development of robust decentralized algorithms in order to coordinate and synchronize the concurrent activities of the agents.

Centralized MAP. A centralized MAP solver is one that either defines a single planning agent or partitions the task among a set of agents $\mathcal{AG}$ while keeping a self-contained execution. As illustrated on Figure 2 (left), the main characteristic of centralized MAP methods is that they solve tasks in a monolithic fashion, executing all the planning agents in a single host. The solver receives the description of the MAP task $\mathcal{T}$ and executes its internal planner processes $\{P_1, \ldots, P_n\}$ that result from partitioning the task in a single machine (only P_1 for single-planner solvers), producing a solution plan Π.

The motivation for choosing a centralized MAP scheme is twofold: 1) the planning process is not based on distributed algorithms, so external communications over the network are not needed to coordinate the planning processes of the agents; and 2) centralized MAP systems can easily reuse robust and efficient single-agent planning technology.

In centralized MAP, we can distinguish solvers that make use of a single planning agent or centralized planner, and solvers that internally partition the MAP task among multiple planning agents or planner processes, without implementing actual software agents:

— **One planning agent:** The development of MAP solvers that fall into the conceptual category of *planning for multiple executors* (see Table II) is based on a single planning agent that works under complete knowledge of the MAP task. TFPOP [Kvarnström 2011], for instance, uses a centralized planner and introduces constraints in the solution plans to properly distribute the actions among the execution agents.

— **Multiple planning agents:** In this approach, the solver partitions the task into multiple planner processes but these processes are not actually implemented as software agents. This approach is followed by ADP [Crosby et al. 2013], which applies an *agentization* procedure in order to obtain a decoupled MAP task from a classical STRIPS-style planning problem. Then, the MAP task is solved in a centralized fashion, determining in each iteration the set of subgoals achievable by one agent from the current state.

Many of the MAP solvers that conceptually fall under the category of *planning by multiple agents* (see Table II), such as MARC [Sreedharan et al. 2015] or MAPR [Borrajo 2013], are implemented by partitioning the task into multiple planning agents. MARC rearranges agents in $\mathcal{AG}$ by grouping them into transformer agents in order to decouple the MAP task as much as possible and then the task is solved in a centralized fashion. MAPR establishes a sequential execution order among the planning agents and uses a centralized planner that progressively and incrementally builds a solution plan according to the pre-established sequence of agents. All in all, MAPR partitions the task among several planning processes while maintaining a single execution thread during the problem resolution.

Distributed MAP. Many of the approaches that conceptually conceive MAP as a process carried out by multiple planning agents (see Table II), are developed on a *distributed* fashion; that is, they implement a group of independent software agents that plan together. By software agent, we refer to a system that 1) makes decisions without any external intervention (autonomy), 2) responds to changes in the environment (reactivity), 3) exhibits goal-directed behaviour by taking the initiative (pro-activeness), and 4) interacts and collaborates with other agents via some communication language in order to achieve its objectives (social ability) [Wooldridge 1997].

In this context, the task information and the planning capabilities are distributed among agents, such that each agent $i \in \mathcal{AG}$ has a different view on the task, $\mathcal{T}^i$. Moreover, unlike centralized solvers, distributed approaches to MAP can be concurrently run in several machines (see Figure 2 (right)). More precisely, given a task $\mathcal{T}$ where $|\mathcal{AG}| = n$, an agent-based solver can be run in up to n machines $\{M_1, \ldots, M_n\}$; that is, each planning agent $i \in \mathcal{AG}$ is independently executed on a machine M_i and receives its view of the task, $\mathcal{T}^i$, as an input.

The focus of the distributed or agent-based approach lies in the coordination of the activities of the different software planning agents in a context in which information is distributed. Since the software agents that compose a distributed MAP solver may be run in different hosts (see Figure 2), having a proper communication infrastructure and message passing protocols is vital for the synchronization of the agents' activities.

Distributed approaches like MAD-A* [Nissim and Brafman 2012] and FMAP [Torreño et al. 2014b] interpret distributed MAP as a multi-agent heuristic search in which agents jointly explore a search tree. In MAD-A*, agents build individual state-based search trees and coordinate their activities by exchanging the states that have relevant information for other agents. Agents in FMAP maintain a synchronized copy of a multi-agent plan-based search tree. At each iteration of the FMAP algorithm, the successor plans individually generated by the agents are exchanged in a privacy-preserving manner.

PSM [Tožička et al. 2015b] is based on planning state machines (PSM); i.e., finite automatons that provide a compact representation of a set of plans. Agents concurrently generate plans, which are then stored into PSMs, until a joint solution plan is found.

All in all, distributed or agent-based MAP solvers design and implement robust distributed algorithm with a special attention on the usage of communication infrastructures in order to allow agent to efficiently cooperate towards the construction of solution.

Distributed approaches to MAP face several challenges, such as the definition of global heuristic functions. Due to the inherently decentralized design of distributed approaches and so the need of privacy-preserving mechanisms, single-agent heuristics are not directly applicable. Consequently, in most distributed MAP approaches agents apply a local evaluation of their plans. In other cases, solvers attempt to come up with global heuristics that take into account the information of the entire task $\mathcal{T}$ but this requires an extensive use of communications among agents (see section 3.3).

3.2. Coordination Strategies and Resolution Schemes

The two principal activities carried out by a MAP solver are planning and coordination. Planning is an inherently individual process and coordination is a collective process. Single-planner MAP solvers, like the ones that conceptually apply planning for multiple agents, leverage the distributed nature of a MAP task and solve the tasks of the execution agents in a unified manner. However, in approaches that use several planning agents, the planning activity is locally attained by each participating agent.

Coordination is defined as a multi-agent decision-making process that allows agents to harmonize their planning activities. Therefore, coordination is only required by MAP solvers that conceptually draw upon multiple planning agents (see right column of Table II). Coordination of agents involves activities such as distributing the MAP task goals among the participants, jointly selecting the next node to expand on a search tree, or combining the agents' local solutions by removing inconsistencies among them. Ultimately, the objective is to synthesize a sound global solution for the MAP task.

The typology of a MAP task often determines the coordination requirements that are necessary for solving the task, where cooperative tasks, for instance, usually demand a stronger coordination effort than collaborative tasks (see section 2.2). Thus, the capability and efficiency of a solver for solving certain MAP tasks is determined by the planning and coordination strategy that governs the solver behaviour. In the literature, we can identify two principal planning and coordination strategies that are followed by the majority of solvers:

— **Unthreaded planning and coordination:** Some solvers define planning and coordination as *unthreaded* or *sequential* activities. In this strategy, each agent i plans for solving its local planning task, $\mathcal{T}^i$, and coordination takes place *before* or/and *after* planning. The unthreaded strategy may involve introducing constraints before planning to guarantee that the agents' local plans are combined into a robust global solu-

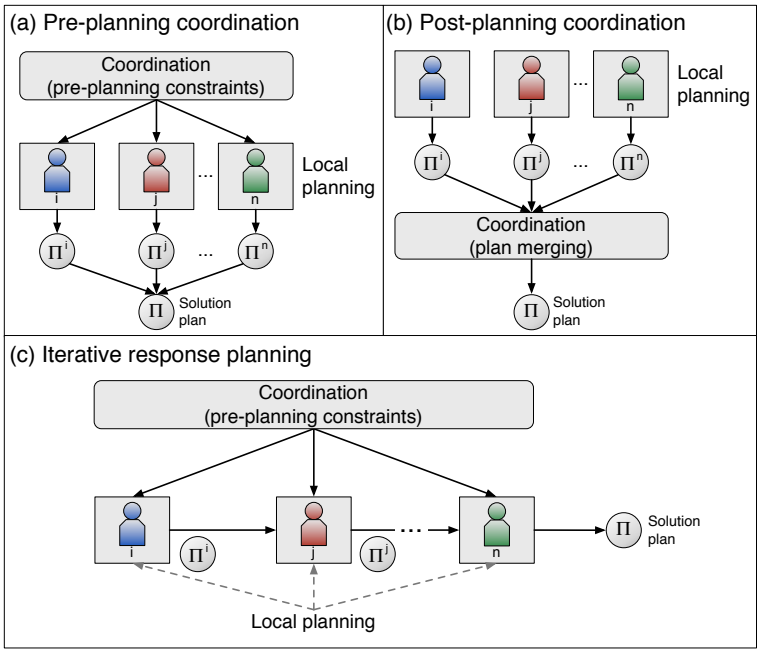

Fig. 3. Resolution schemes in unthreaded planning and coordination

tion, or combining the agents' local solution plans into a global plan in post-planning time (see Figures 3 (a) and 3 (b)).

An unthreaded solver can also follow an *iterative response* scheme, in which the various agents perform planning sequentially. An agent i coordinates with the next agent in the sequence, j, by communicating its local solution plan Π^i to j. In turn, agent j uses Π^i to generate an extended solution plan, Π^j, which is then communicated to the following agent and so on (see Figure 3 (c)).

— **Interleaved planning and coordination:** Many approaches to MAP combine or *interleave* planning and coordination, such that coordination is alternated with planning episodes (see Figure 4). In the interleaved strategy, agents carry out a synchronized group planning activity by continuously coordinating with each other during planning.

The following subsections provide a formal analysis of the resolution schemes used in these two approaches to planning and coordination.

3.2.1. Unthreaded Planning and Coordination. MAP solvers that apply an unthreaded planning and coordination strategy conceive both activities as separate black boxes. Under this strategy, each agent $i \in \mathcal{AG}$ synthesizes a solution plan to its local task $\mathcal{T}^i$ and coordination is applied either before or after planning in order to guarantee the combination of the local plans into a robust global solution that attains the task goals.

Pre-planning coordination. In the pre-planning coordination scheme, the solver defines the constraints that ensure that the agents' local solutions can be combined into a consistent solution plan for $\mathcal{T}$ (see Figure 3 (a)).

ADP [Crosby et al. 2013] follows this scheme by applying an agentization procedure that distributes a STRIPS planning task into several planning agents (see Table II).

More precisely, ADP is a fully automated process that inspects the multi-agent nature of the planning task and calculates an agent decomposition that results in a set of n decoupled local tasks. ADP applies a centralized, sequential, total-order planning algorithm that takes advantage of this multi-agent decomposition and yields a solution for the original STRIPS task. Since the task is broken down into several decoupled and individual sub-problems, the solution plans to these sub-problems are consistent to each other and post-planning coordination is not required. Therefore, ADP uses a divide-and-conquer approach to MAP, where the local sub-tasks are not only smaller and easier to solve than the original single-agent task but also they are independent to each other, thus ensuring that the local sub-plans can be seamlessly combined into a global solution for the MAP task $\mathcal{T}$.

All in all, the purpose of pre-planning coordination is the establishment of robust constraints to generate independent or decoupled local sub-tasks. This way, it can be guaranteed that the local plans generated by the agents are seamlessly combined into a sound global solution that attains the MAP task, thus avoiding the use of plan merging techniques at post-planning time.

Post-planning coordination. Other MAP solvers based on unthreaded planning and coordination put the coordination emphasis *after* planning. In this case, the objective is to *merge* the solutions to the agents' local tasks $\{\mathcal{T}^1, \ldots, \mathcal{T}^n\}$ into a global plan that attains the goals $\mathcal{G}$ of the task $\mathcal{T}$ by removing the inconsistencies among these local solutions (see Figure 3 (b)).

For example, PMR (Plan Merger by Reuse) [Luis and Borrajo 2014] allows planning agents to build plans that solve their local subtasks, and then, it concatenates the agents' local plans, $\{\Pi^1, \ldots, \Pi^n\}$, into a global solution plan Π. Given that the concatenation of plans does not always result in a sound plan, PMR executes the LPG-ADAPT planner [Fox et al. 2006], taking as an input the possibly invalid plan, to synthesize a valid global solution.

Other approaches based on post-planning coordination address MAP through the intersection of finite automata that represent the local plans of the agents. This strategy was firstly introduced by Distoplan [Fabre et al. 2010], a factored-planning approach that uses a message passing algorithm to communicate the edges of the graph that connects the components of the MAP task. PSM [Tožička et al. 2015b] draws upon a set of finite automata, called Planning State Machines (PSM), where each automaton represents the set of local solution plans of a given agent. In an iteration of the PSM procedure, each agent $i \in \mathcal{AG}$ generates a solution plan for its local task $\mathcal{T}^i$ and incorporates it to its associated PSM. Then, agents calculate and exchange the pubic projection of their PSMs, until a global solution for the task $\mathcal{T}$ is found.

Iterative response planning. A third resolution scheme that falls within the unthreaded planning and coordination strategy consists in executing planning agents sequentially, so that coordination is applied after each agent completes its planning activity. In the *iterative response planning* scheme, firstly introduced by DPGM [Pellier 2010], an agent receives the previous agent's local solution and a set of constraints for coordination purposes, and *responds* by building up a solution for its local subtask on top of the received plan. This way, the solution plan is incrementally synthesized (see Figure 3 (c)).

Multi-Agent Planning by Reuse (MAPR) [Borrajo 2013] is an iterative response solver based on *goal allocation*. MAPR distributes the task goals $\mathcal{G}$ among the participating agents before planning, such that an agent i is assigned a subset of goals $\mathcal{G}^i \subset \mathcal{G}$. A goal in $\mathcal{G}$ is assigned to only one planning agent, i.e., $\bigcap_{i \in \mathcal{AG}} \mathcal{G}^i = \emptyset$. As a result, the

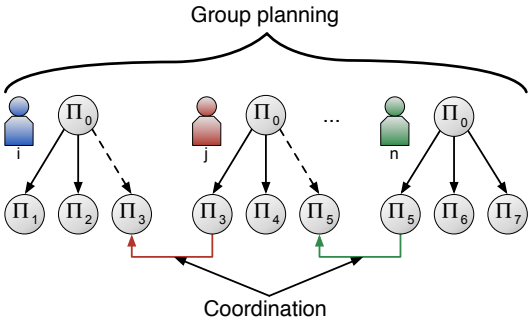

Fig. 4. Multi-agent search in interleaved planning and coordination

local view of the task received by an agent $i \in \mathcal{AG}$ is of the form $\mathcal{T}^i = \langle \mathcal{V}^i, \mathcal{A}^i, \mathcal{I}^i, \mathcal{G}^i \rangle$. Additionally, agents are sequentially arranged, thus defining the order in which the iterative response scheme must be carried out.

Unlike most unthreaded solvers, which apply a local state-based search to incrementally synthesize the solution plans, DPGM [Pellier 2010] uses a hybrid resolution scheme that extracts plans from distributed planning graphs through a CSP solver. The first agent in the sequence proposes a plan as well as a set of coordination constraints, and the following agent obtains a plan that complies with the constraints defined by the first agents. DPGM backtracks to the previous agent in case that a solution cannot be found.

To sum up, planning and coordination are conceived as two separately and clearly identified stages in unthreaded solvers. In some cases, constraints are applied at pre-planning time to ensure that the local solution plans are consistent. These constraints involve distributing or balancing the MAP task, allocating goals to the agents or establishing the execution order of the planning activities. Coordination can also occur at post-planning time in the form of a plan merging process that removes inconsistencies among the agents' local plans, thus obtaining a global solution plan for the MAP task $\mathcal{T}$.

In iterative response approaches, coordination occurs both before and after planning. In pre-planning time, the solver introduces a set of constraints, either in the form of coordination constraints [Pellier 2010] or goal assignments [Borrajo 2013]. Once an agent i completes its planning activity, it coordinates with the next agent in the sequence, j, by communicating its local solution Π^i, which is the input of agent j.

Unthreaded planning and coordination is an efficient strategy that effectively minimizes the communication needs during planning, since agents do not interact to each other while planning. Moreover, the unthreaded strategy is especially efficient at solving collaborative tasks that do not require a high coordination effort.

However, this strategy presents several limitations when solving cooperative tasks. In an unthreaded solver, agents are unable to discover and address the cooperation demands of other agents because the underlying resolution scheme relies upon autonomous agents that are capable of solving their assigned tasks by themselves. The needs of cooperation that arise when solving cooperative goals are hard to discover in pre-planning time, and the plan merging techniques are designed only to fix inconsistencies among local plans, rather than repairing the plans to satisfy the inter-agent coordination needs. A higher coordination effort is thus required to efficiently solve cooperative goals.

3.2.2. Interleaved Planning and Coordination. A large number of MAP techniques *interleave* the activities of planning and coordination. This strategy is characterized by the application of *group planning*; i.e., rather than synthesizing a local solution individually, agents explore the search space jointly in order to synthesize a solution plan. Under this strategy, agents continuously coordinate with each other to communicate their findings, thus effectively intertwining planning and coordination.

Most interleaved MAP solvers, such as the MAFS [Nissim and Brafman 2012] and FMAP [Torreño et al. 2014b] families, commonly rely on a *coordinated multi-agent search* resolution scheme, in which nodes of the search space are contributed by several agents (see Figure 4). This resolution scheme involves the selection of a node for expansion, an individual expansion of the node by each agent (planning) and an exchange of the resulting successor nodes among the agents. Agents explore the search space until a solution is found, alternating phases of planning and coordination.

Different forms of coordination are applicable in an interleaved planning and coordination strategy. In FMAP [Torreño et al. 2014b], agents perform a plan-based group search. Agents maintain a common open list of plans, whose quality is assessed through the global heuristic function h_{DTG} (see section 3.4). First, agents select the most promising open node according to the heuristic criterion and each agent i expands the selected node using its actions $\mathcal{A}^i$. Next, agents evaluate and exchange all the successor plans they have generated. If an agent i shares a plan with an agent j, it occludes the information of the plan that is included in $\mathcal{T}^i$ but not in $\mathcal{T}^j$. As a result, each agent holds an equivalent copy of the search tree that preserves its privacy constraints from the rest of participating agents.

In MAFS [Nissim and Brafman 2012], each agent $i \in \mathcal{AG}$ maintains a separate search space and an independent open list of states. The search process is carried out simultaneously by the planning agents in $\mathcal{AG}$. An agent i selects the next state in its open list to expand, S, according to the application of a local heuristic estimate, and generates a set of successors over S using its own actions only, $\mathcal{A}^i$. Out of all the successor nodes, agents only share the states that are relevant to other agents. A state S generated by an agent i is relevant to another agent j if j has at least one action whose preconditions are in S, and the action which generated S is in $\mathcal{A}^j$. The search process carries on until a state S_g, where $\mathcal{G} \subseteq S_g$, is found.

The interleaved planning and coordination strategy is very suitable for solving complex tasks that involve cooperative goals and a high coordination effort. By using this strategy, agents learn the cooperation requirements of other participants during the construction of the plan and can immediately address them. Hence, this strategy allows agents to efficiently address cooperative goals.

The main drawback of the interleaved strategy is the high cost of communications in a distributed MAP setting because alternating planning and coordination usually entails exchanging a high number of messages in order to continuously coordinate the agents' activities.

3.3. Heuristic Search

Most MAP solvers in the literature resort to heuristic search during the resolution of the task. In general, the single planning entity ($|\mathcal{AG}| = 1$) of solvers that plan *for* multiple executors (see Table II) has a complete knowledge of the MAP task $\mathcal{T}$, so these approaches apply heuristic evaluation using the *global* information of the task, equivalently to single-agent planners.

However, in solvers that feature multiple planning agents, i.e., $|\mathcal{AG}| > 1$, a given agent i has usually a limited view of the task, $\mathcal{T}^i$, and no agent has access to the global information of the task, $\mathcal{T}$. Consequently, heuristic evaluation is applied differently in this case.

In this setting, one can distinguish between heuristics applied *locally* or *globally*. In the former, each agent $i \in \mathcal{AG}$ applies the heuristic evaluation from its local perspective, that is, from $\mathcal{T}^i$. The advantage of this approach is its simplicity, since no interaction with other agents is required to perform the evaluation. However, the accuracy of the estimates is compromised by the limited view of the agent; in general, local heuristics are less accurate than heuristics which take into consideration the complete information of the MAP task $\mathcal{T}$.

In contrast, a global heuristic is defined as the distributed application of a heuristic function by several entities with limited information of the MAP task in a privacy-preserving context [Torreño et al. 2015]. The development of distributed heuristics constitutes one of the current challenges in decentralized MAP, because multi-agent scenarios present additional features that make heuristic evaluation an arduous task [Nissim and Brafman 2012]:

— As previously mentioned, solvers that draw upon multiple planning agents do not feature an entity that has a complete knowledge of the map task $\mathcal{T}$. Additionally, solvers based on a distributed implementation require robust communication protocols among the agents in order to calculate estimates for the overall task.
— For the MAP approaches that preserve agent's privacy, the communication protocol must guarantee that estimates are computed without disclosing sensitive private information.

The local or global application of a heuristic search may be determined by the characteristics of the resolution scheme of the MAP solver. Particularly, agents of solvers based on *unthreaded* planning and coordination synthesize local solutions individually, which forces the use of local heuristic functions. For instance, MAPR [Borrajo 2013] applies an iterative response planning scheme that performs goal allocation to the participants and solves the agents' subtasks sequentially. Thereby, the heuristic functions used by MAPR, h_{FF} [Hoffmann and Nebel 2001] and h_{Land} [Richter and Westphal 2010], are locally computed.

Local heuristic search has also been strengthen by some *interleaved* MAP solvers. Agents in MAFS and MAD-A* [Nissim and Brafman 2014] generate and evaluate search states locally. An agent i shares a state S only if it is relevant to other planning entities, along with i's local heuristic estimate of the state S. When an agent j receives S from agent i, j performs its local evaluation of S. Then, depending on the characteristics of the applied heuristic functions, agent j assigns a heuristic value to S, which can be the received estimate, its own evaluation estimate or a combination of both. In [Nissim and Brafman 2012], authors test MAD-A* with two different optimal heuristics, LM-Cut [Helmert and Domshlak 2009] and merge-and-shrink [Helmert et al. 2007], both locally applied by each agent. Despite only local heuristics are used, MAD-A* is proven to be cost-optimal when using h_{FF} and h_{Land}.

Unlike unthreaded solvers, the interleaved planning and coordination strategy makes it possible to accommodate global heuristic functions. In this case, agents perform group planning and synthesize together a solution for the MAP task, thus enabling the use of global heuristics.

GPPP [Maliah et al. 2014] introduces a distributed version of a privacy-preserving landmarks extraction algorithm for MAP. The heuristic estimate of a state is obtained as the sum of the local estimates of each agent. GPPP outperforms MAFS thanks to the accurate estimates provided by this landmark-based heuristic.

FMAP [Torreño et al. 2014b] uses a global state-based heuristic function, h_{DTG}, to evaluate partial-order plans. h_{DTG} draws upon the information of the *Domain Transition Graphs* [Helmert 2004] associated to the state variables of the MAP task. h_{DTG}

computes a relaxed plan between the *frontier state* [Benton et al. 2012] of a plan and the task goals $\mathcal{G}$. The relaxed plan is calculated by finding in the DTG of each variable the shortest path between the value of the variable in the frontier state and the value of the state variable in the goal state.

MH-FMAP [Torreño et al. 2015], the latest version of FMAP, introduces a multi-heuristic search strategy for MAP inspired by the heuristic alternation mechanism of Fast Downward (FD) [Helmert 2006]. Agents in MH-FMAP select the next plan to expand alternatively from two different search queues: the first one sorts the plans by means of h_{DTG}, while the second one keeps track of the *preferred successors* [Torreño et al. 2015], which are evaluated with another global heuristic, h_{Land}. This heuristic function counts the number of landmarks that are not satisfied in a plan.

The work in [Štolba and Komenda 2013] adapts the well-known h_{FF} heuristic function to a multi-agent context by means of *distributed Relaxed Planning Graphs* (dis-RPGs) [Zhang et al. 2007]. The multi-agent version of h_{FF} is reported to yield the same quality than the single-agent h_{FF} [Hoffmann and Nebel 2001]. However, we must note that the large number of agent communications needed for building and exploring the dis-RPG, render the calculation of the multi-agent version of h_{FF} a very costly procedure.

Authors in [Štolba and Komenda 2014] present a distributed implementation of several relaxed heuristics (a relaxed version of h_{FF}, h_{add} and h_{max}). Instead of a dis-RPG, authors devise a more compact structure, the *exploration queue*, that significantly reduces the number of messages exchanged among agents. This multi-agent version of the h_{FF} heuristic, however, is not as accurate as the single-agent function.

Finally, the work in [Štolba et al. 2015] introduces a multi-agent global version of the admissible heuristic function LM-Cut that is proven to obtain estimates of the same quality than the single-agent LM-Cut function. Granting this property increases the computational cost of the heuristic, but the additional accuracy of the multi-agent LM-Cut function compensates this disadvantage.

In conclusion, heuristic search in MAP, and most notably, the development of global heuristic functions in a distributed context, constitutes one of the main challenges of the MAP research community. The aforementioned works prove the potential of the development and combination of global heuristics towards scaling up the performance of MAP solvers.

3.4. Privacy

The preservation of agents' sensitive information, or *privacy*, is one of the basic aspects that must be enforced in MAP. The importance of privacy is illustrated in Examples 2.4 and 2.5 of section 2.2, which include two different agents, $ta1$ and $ta2$, that represent two transport agencies. Whereas both agents collaborate in these particular tasks, it is unlikely that they are willing to reveal sensitive information of their internal procedures to a potential competitor. These two examples clearly emphasize the role of privacy in distributed MAP.

Privacy in MAP has been fairly neglected and under-represented in the literature. Moreover, formal treatment of privacy is even more scarce. Early MAP approaches neglected privacy issues and focused instead on information distribution, effective computation or task decomposition, among other aspects.

One of the first attempts to come up with a formal privacy model in MAP can be found in [van der Krogt 2007], where authors quantify privacy in terms of the Shannon's information theory [Shannon 1948]. More precisely, authors establish a notion of uncertainty with respect to plans and provide a measure of privacy loss in terms of the data uncovered by the agents along the planning process. Unfortunately, this

measure is not general enough to capture details such as heuristic computation. Nevertheless, quantification of privacy is an important step in MAP, as it is in distributed CSP [Faltings et al. 2008]. A more recent work, also based on Shannon's information theory [Štolba et al. 2016c], quantifies privacy leakage for MA-STRIPS according to the reduction of the number of possible transition systems caused by the revealed information. The main sources of privacy leakage are identified, but not experimentally evaluated.

The MA-STRIPS solver [Brafman and Domshlak 2008] established the basic privacy guidelines in the literature, which have been followed by many MAP solvers ever since. MA-STRIPS infers the private information directly from the task structure and classifies propositions as either public or internal, where a proposition is internal to an agent $i \in \mathcal{AG}$ if the proposition is only achieved or deleted by i. According to this notion, actions are partitioned into internal and public, and an internal action only contains internal preconditions and effects.

The privacy properties of a MAP solver can be studied from different standpoints:

— **Modelling of private information:** Many approaches to MAP, including MA-STRIPS, apply a form of *induced* privacy; that is, the solver directly infers private information from the task structure. Recent solvers, such as FMAP [Torreño et al. 2014b], adopt a more general approach by *imposing* privacy as a part of the task description.
— **Information sharing:** MA-STRIPS defines public information as known to all the participating agents, while *internal* or *private* information is known to a single agent only. This definition is too restrictive for contexts that require information to be shared among a subset of agents. More recent approaches introduce the notion of *subset privacy*, by which public information is accessible to either all the participants or just a subset of agents [Torreño et al. 2012; Bonisoli et al. 2014].
— **Privacy practical guarantees:** Recent studies discuss and formally analyze the privacy practical properties of a MAP solver. The work in [Brafman 2015] defines two levels of privacy guarantees: a *weak* privacy level allows an agent to infer or deduce private information that is not explicitly transmitted by other agents; and a *strong* privacy level guarantees that no agent is able to deduce private data from others. The work in [Shani et al. 2016] introduces a third privacy level, *object cardinality* privacy, in which an agent cannot deduce the number of objects of a certain type managed by another agent.

The next subsections analyze the privacy models followed by the MAP solvers in the literature according to the three aforementioned criteria.

3.4.1. Modelling of Private Information. Modelling of private information is highly related to the language used to describe the MAP task as some specification languages enable a explicit modelling of privacy and others do not.

Early approaches to MAP, such as MA-STRIPS [Brafman and Domshlak 2008], manage a notion of induced privacy. Since the *MA-STRIPS* language does not explicitly model private information, MA-STRIPS infers the agents' private data from the structure of the MAP task. Given an agent $i \in \mathcal{AG}$ and a piece of information $p^i \in \mathcal{T}^i$, p^i is identified as private if $\forall_{j \in \mathcal{AG} | j \neq i} p^i \notin \mathcal{T}^j$; that is, the data contained in p^i are known to i and ignored by the rest of participants of the MAP task $\mathcal{T}$.

One of the most well-known refinements of the MA-STRIPS model extends the notion of induced privacy to a more general *imposed privacy* scheme, in which the private or shareable information is explicitly stated in the task description. The imposed privacy scheme is first introduced in FMAP [Torreño et al. 2014b], which incorporates a

`:shared-data` construct to define the agent's shareable information in the description language (see section 2.3).

MA-PDDL [Kovacs 2012], the language used in the CoDMAP competition [Štolba et al. 2016b], has been adopted by many state-of-the-art solvers (see Appendix A). *MA-PDDL* also follows the trend of imposed privacy and allows the designer to define the private elements of an agent's task through the `:private` identifier.

In general, both induced and imposed privacy schemes are commonly applied by current MAP solvers. The induced privacy scheme enables to automatically identify the natural private elements of a MAP task. The imposed privacy scheme, however, offers the task designer more control and flexibility to define privacy, which is a helpful tool in contexts where agents want to occlude sensitive data that would be shared otherwise.

3.4.2. Information Sharing. Privacy preservation models present several differences regarding how many agents can share a particular piece of information. In general, there are two information sharing models, namely the MA-STRIPS scheme and the *subset privacy* model.

MA-STRIPS. The MA-STRIPS privacy model [Brafman and Domshlak 2008] can be considered as the simplest model of information sharing, defining as public the data that are shared among all the participating agents in $\mathcal{AG}$. In this privacy model, a piece of information is known to either all the participants in $\mathcal{AG}$ or to a single agent.

In MA-STRIPS, a proposition or fluent $\langle v = d \rangle$ is defined as *internal* or *private* to an agent $i \in \mathcal{AG}$ if $\langle v = d \rangle$ is only used and affected by the actions of $\mathcal{A}^i$. However, if the fluent $\langle v = d \rangle$ is also in the preconditions and/or effects of some action $\alpha \in \mathcal{A}^j$, where $j \in \mathcal{AG}$ and $j \neq i$, then the fluent $\langle v = d \rangle$ is publicly accessible to all the agents in $\mathcal{AG}$.

Given an action $\alpha \in \mathcal{A}^i$, if the fluents that appear in the preconditions and effects of α are private to i, then α is said to be public and is known to all the participants in the task. If α includes both public and internal preconditions and effects, agents share instead α_p, the *public projection* of α, an abstraction that contains only the public part of α.

The simplistic assumptions of MA-STRIPS, by which a public proposition or action is shared among all the participating agents, may not be allowed in applications where some information needs to be shared only by some planning agents in $\mathcal{AG}$. For instance, let us assume that agent $ta2$ in Examples 2.4 and 2.5 wants to communicate the factory agent ft that its truck $t2$ is currently placed in ft, i.e. $pos(t2) = ft$, but it wants to occlude this fact from agent $ta1$. In this case, the information sharing policy used by MA-STRIPS would not be expressive enough to model this situation.

Subset privacy. This model, introduced in [Torreño et al. 2014a; Bonisoli et al. 2014], generalizes the MA-STRIPS scheme by establishing pairwise privacy. This model defines a piece of information as either private to a single agent, publicly accessible to all the agents in $\mathcal{AG}$ or known to a *subset of agents* $\{i, \ldots, m\} \subset \mathcal{AG}$. This approach is useful in applications where agents wish to conceal some information from certain agents.

For instance, agent $ta2$ in Examples 2.4 and 2.5 notifies the factory agent ft whenever it reaches the fact $pos(t2) = ft$. This fluent indicates that the truck $t2$ is placed at the factory ft, a location that is known to both $ta2$ and ft. However, the MA-STRIPS model is not a suitable approach if $ta2$ does not want to inform agents that work on the same business area, such as $ta1$, of its transportation routes and resources. The more appropriate subset privacy approach allows $ta2$ to effectively hide sensitive information from agent $ta1$ while sharing it with the factory ft.

To sum up, the subset privacy model is a more flexible information sharing model compared to the more conservative and limited approach followed by MA-STRIPS, allowing to represent more complex and realistic situations concerning information sharing.

3.4.3. Privacy Practical Guarantees. Recent studies in the literature, devoted to formally discuss the privacy practical guidelines of MAP solvers, conclude that the privacy schemes of some solvers may allow agents to infer private information of other agents through the transmitted data. According to these studies [Nissim and Brafman 2014; Shani et al. 2016], it is possible to establish a four-level taxonomy to classify the actual practical privacy level of MAP solvers. The four levels of the taxonomy, from the least to the most secure one, are: *no privacy*, *weak privacy*, *object cardinality privacy* and *strong privacy*.

No privacy. Many early approaches to MAP neglected privacy issues and applied a no privacy scheme. Among such solvers is the first MA-STRIPS-based planner, Planning First [Nissim et al. 2010] and the agent-decomposition planner ADP [Crosby et al. 2013]. Moreover, approaches other than MA-STRIPS-based planners, such as μ-SATPLAN [Dimopoulos et al. 2012], A# [Jezequel and Fabre 2012], DPGM [Pellier 2010] or GPGP [Decker and Lesser 1992], do not consider either privacy as a relevant issue.

Weak privacy. A MAP system is said to be weakly privacy-preserving if agents do not explicitly communicate their private information to other agents at execution time [Brafman 2015]. A weak privacy level is accomplished by either *obfuscating* (encrypting) or *occluding* the private information before transmitting it, so that agents only reveal the public projection of their actions. In a weak privacy setting, agents may infer private variables, values, and action preconditions and effects of other agents through the information exchanged during the MAP resolution process.

Obfuscating or encrypting the private elements of the planning task is a common mechanism to implement weak privacy. In MAPR [Borrajo 2013], PMR [Luis and Borrajo 2014] and CMAP [Borrajo and Fernández 2015], when an agent transmits a plan, it obfuscates or encrypts the private propositions in the preconditions, effects, initial state and goals in order to preserve its sensitive information to the receivers. Proposition names are encrypted, but the number and unique identity of preconditions and effects of the actions are retained, so agents are able to reconstruct the complete isomorphic image of their tasks. Obfuscation is an appropriate technique when agents wish to conceal the meaning of predicates and actions (for instance, agents in Examples 2.4 and 2.5 may want to preserve private information such as the links among the locations in their working areas or the position of a truck whenever it is placed at a private location).

In MAFS [Nissim and Brafman 2014], MADLA [Štolba and Komenda 2015], MAPlan [Fišer et al. 2015] and GPPP [Maliah et al. 2014], agents exchange the relevant states to each other in order during search (see section 3.2.2). These approaches follow an obfuscation mechanism as agents encrypt the values of their private variables when sharing a state with other participants. The obfuscation technique used by the aforementioned approaches allows agents to infer some private information, and therefore, these solvers grant only weak privacy.

In PSM [Tožička et al. 2014], agents share only public projections of the Planning State Machines (PSMs), which encode a set of plans that attain the public part of the MAP task. However, given a PSM Γ_1 with two states $\{a, x\}$ and $\{a, y\}$, where a represents public facts and x and y are sets of private facts, the public projection of Γ_1 does not replace $\{a, x\}$ and $\{a, y\}$ by a single public state $\{a\}$ because otherwise, the topology of this public projection would be altered and the resulting PSM would mis-

takenly accept public projections of plans that are not actually accepted by Γ_1 [Tožička et al. 2015b]. To address this limitation, authors use an encryption mechanism that replaces x and y by *integer marks*, so that the PSM public projection includes two different states $\{a\}'0$ and $\{a\}'1$.

Other weak privacy-preserving solvers in the literature *occlude* the agents' private information, instead of sharing obfuscated data. In the FMAP [Torreño et al. 2014b] family of solvers, which also includes MAP-POP [Torreño et al. 2012; 2014a] and MH-FMAP [Torreño et al. 2015], agents only exchange the public projection of the actions of the partial-order plans they have generated, occluding the private part of their plans. Given a fluent $\langle v = d \rangle$, an agent i shares the fluent with an agent j if $v \in \mathcal{V}^j$ and $d \in \mathcal{D}_v^j$, or occludes it if $v \notin \mathcal{V}^j$. In case that $v \in \mathcal{V}^j$ but $d \notin \mathcal{D}_v^j$, agent i shares instead $\langle v = \perp \rangle$, where $\perp$ represents the undefined value [Torreño et al. 2014b]. This occlusion mechanism is also used in the global heuristic function h_{DTG} applied in both FMAP and MH-FMAP.

MH-FMAP combines h_{DTG} and the landmark-based heuristic h_{Land}. The implementation of h_{Land} uses an encryption technique to maintain privacy during the joint construction of the landmarks graph [Torreño et al. 2015]. Therefore, MH-FMAP can be regarded as a hybrid approach that combines both occlusion and obfuscation techniques.

In general, most global heuristic functions in MAP aim for weak privacy by obfuscating or occluding private information [Torreño et al. 2014a; 2015; Štolba and Komenda 2013; 2014; Štolba et al. 2015; Maliah et al. 2014; 2015]. Recent works suggest the use of additive heuristics, which are computed by adding together the projected estimates of the agents, thus avoiding the costly communication needed by privacy-preserving global heuristic estimates [Štolba et al. 2016a].

Object cardinality privacy. Recently, the DPP planner [Shani et al. 2016] introduces a new level of privacy named object cardinality privacy.

A MAP algorithm preserves object cardinality privacy if, given an agent i and a type t, the cardinality of i's private objects of type t cannot be inferred by other agents from the information they receive [Shani et al. 2016]. In other words, this level of privacy strongly preserves the number of objects of a given type t of an agent i, thus representing a middle ground between the weak and strong privacy settings.

Hiding the cardinality of private objects is motivated by real-world scenarios. Consider, for example, the logistics task in Examples 2.4 and 2.5. One can assume that the transport agencies that take part in the MAP task, $ta1$ and $ta2$, know that packages are delivered using trucks. However, it is likely that each agent would like to hide sensitive information related to their capabilities, such as the number of trucks or transportation routes.

Strong privacy. One limitation of the weak privacy level is that it allows agents to infer private information from the public data they receive. Let us consider again the Example 2.4 of section 2.2; as detailed in Table I, transport agency $ta1$ does not disclose the existence or the location of its truck $t1$ to agent $ta2$; i.e., the $pos(t1)$ variable is private to agent $ta1$. Moreover, $ta1$ only informs $ta2$ about the position of the package p whenever it is placed at the public location sf; that is, $at(p) = sf$. Assuming that one state contains $pos(t1) = sf$ and $at(p) = sf$, and that $t1$ loads the package p, the descendent state will comprise $at(p) = t1$ and this fluent will not be transmitted to $ta2$, as it happens in MAFS [Nissim and Brafman 2014]. Agent $ta2$ will thus notice that the package p disappears from sf in the descendent state and, consequently, it might deduce that agent $ta1$ has a resource that enables it to change the location of the package p.

Secure-MAFS [Brafman 2015], an upgraded version of MAFS [Nissim and Brafman 2014], is the first and only MAP solver in the literature that provides strong privacy guarantees. In Secure-MAFS, two states that only differ in their private elements are not communicated to other agents in order to prevent them from deducing information through the non-private or public part of the state (e.g., it would be possible to infer private information such that the state of an agent has not changed by observing that the non-private information of two consecutive states has not changed). Secure-MAFS is proved to guarantee strong privacy for a sub-class of tasks based on the well-known *logistics* domain.

A MAP algorithm strongly preserves privacy if none of the agents are able to infer a private element of an agent's task from the public information it receives during planning. In other words, in a strong privacy-preserving solver, private information cannot be deduced from the information gained.

In order to guarantee strong privacy it is necessary to consider several additional factors such as the nature of the communication channel used by the participants (synchronous, asynchronous, lossy) or the computational power of the agents.

In this section, we have seen that weak privacy is easy to achieve through obfuscation or encryption of the private information, but provides little security. On the other hand, whereas strong privacy significantly improves security, it has not been implemented yet by any MAP solver (Secure-MAFS is simply a theoretical proposal) and its complexity has not been studied yet in the literature. Additionally, object cardinality privacy accounts for a middle ground between the weak and strong privacy levels. In general, the vast majority of the state-of-the-art MAP methods can be classified under the no privacy and weak privacy levels: the earlier approaches to MAP do not consider privacy at all, while most of the recent proposals, which claim to be privacy preserving, implement a form of weak privacy, resorting in most cases to *obfuscation* to occlude the private information in the public projection of the actions and states.

4. DISTRIBUTED AND MULTI-AGENT PLANNING SYSTEMS TAXONOMY

The existing works in MAP cover a wide range of different resolution schemes. As discussed in section 3, one can identify several aspects that determine the features of a MAP solvers, such as problem-solving *distribution*, *coordination*, *heuristic search* or *privacy preservation*. This section presents an in-depth taxonomy that reviews the most significant approaches to MAP in the literature (see Table 3.4.3), analyzing their main features and comparing their similarities and differences. The MAP solvers included in the taxonomy are arranged according to the coordination strategy they implement, an aspect that ultimately determines the resolution scheme followed by an approach. Section 4.1 discusses the MAP solvers that implement an unthreaded approach to planning and coordination, while section 4.2 reviews the interleaved approaches to MAP.

4.1. Unthreaded Planning and Coordination MAP Solvers

MAP solvers that follow an unthreaded planning and coordination strategy understand both activities as separate black boxes, where each activity takes place at once either before or after the other. As listed in Table 3.4.3, unthreaded solvers typically use a particular algorithm for coordination (in most cases, a combinatorial optimization or satisfiability problem solver) and another one for local planning (generally, a state-of-the-art single-agent planner).

Table III. Summary of the state-of-the-art MAP solvers and their features. For unthreaded solvers, the resolution strategies are listed in form of pairs "coordination strategy" & "local planning technique".

MAP Solver	Coord.	Dist.	Resolution strategy	Heuristic	Privacy
Planning First [Brafman and Domshlak 2008]	UT	C	Post-planning coordination via DisCSP & heuristic forward search (FF)	–	N
DPGM [Pellier 2010]	UT	C	Iterative response planning & GraphPlan + CSP plan extraction	–	N
μ-SATPLAN [Dimopoulos et al. 2012]	UT	C	Pre-planning goal allocation → iterative response planning & SAT	–	N
MAPR [Borrajo 2013]	UT	C	Pre-planning goal allocation → iterative response planning → solution plan parallelization & heuristic forward search (LAMA)	L	W
PMR [Luis and Borrajo 2014]	UT	C	Pre-planning goal allocation → Plan merging → plan repair → solution plan parallelization & heuristic forward search (LAMA)	L	W
CMAP [Borrajo and Fernández 2015]	UT	C	Pre-planning goal allocation → task mapping into single-agent task → solution plan parallelization & heuristic forward search (LAMA)	G	W
MAP-LAPKT [Muise et al. 2015]	UT	C	Task mapping into single-agent task & heuristic forward search (LAPKT)	G	W
MARC [Sreedharan et al. 2015]	UT	C	Task mapping into transformer agent task & planning via FD or IBACOP → solution plan translation into original MAP task	G	W
ADP [Crosby et al. 2013]	UT	C	Automated task agentization & heuristic forward search (FD)	L	N
Distoplan [Fabre et al. 2010]	UT	C	Message passing algorithm & Finite Automata	–	N
A# [Jezequel and Fabre 2012]	UT	C	Asynchronous communication mechanism & A* heuristic forward search	G	N
PSM [Tožička et al. 2015b]	UT	D	Intersection of Finite Automata & heuristic forward search (LAMA) → Finite Automata	G	W
DPP [Shani et al. 2016]	UT	C	Synthesis of high-level plan over DP projection (FD) & heuristic forward search (FF)	L	OC
TFPOP [Kvarnström 2011]	UT	C	Forward-chaining partial-order planning & synthesis of agent-specific thread of actions	–	N
MAP-POP [Torreño et al. 2014a]	IL	D	Multi-agent A* heuristic search via backward POP	G	W
FMAP [Torreño et al. 2014b]	IL	D	Multi-agent A* heuristic search via forward POP	G	W
MH-FMAP [Torreño et al. 2015]	IL	D	Multi-agent A* multi-heuristic search via forward POP	G	W
MADLA [Štolba and Komenda 2014]	IL	C	Multi-agent multi-heuristic state-based search	G/L	W
MAFS [Nissim and Brafman 2012]	IL	D	Multi-agent heuristic forward search	L	W
MAD-A* [Nissim and Brafman 2012]	IL	D	Multi-agent A* heuristic forward search	L	W
Secure-MAFS [Brafman 2015]	IL	C	Multi-agent heuristic forward search	L	S
GPPP [Maliah et al. 2016]	IL	C	Multi-agent heuristic forward search (relaxed, subgoals)	G	W
MAPlan [Fišer et al. 2015]	IL	D	Multi-agent heuristic forward search	G/L	W

Distribution: C - centralized, D - distributed (subsumes centralized)
Coordination: UT - unthreaded, IL - interleaved
Privacy: N - no privacy, W - weak privacy, OC - object cardinality privacy, S - strong privacy
Heuristic: L - local, G - global

ACM Computing Surveys, Vol. 0, No. 0, Article 0, Publication date: 2016.

Planning First, 2008 (implemented in 2010). Planning First [Nissim et al. 2010] is the first MAP solver that draws upon the fundamentals of the MA-STRIPS approach [Brafman and Domshlak 2008]. Planning First is an early representative of the unthreaded coordination strategy that inspired the development of many subsequent solvers, which are presented in the next paragraphs.

Planning First aims at efficiently solving loosely-coupled *MA-STRIPS* tasks (see section 2.3). It generates a local plan for each agent in a centralized fashion by means of the FF planner [Hoffmann and Nebel 2001], and coordinates the local plans through a distributed Constraint Satisfaction Problem (DisCSP) solver to come up with a global solution plan. More precisely, Planning First distributes the MAP task among the agents and identifies the coordination points of the task as the actions whose application affects other agents. Then, it defines a DisCSP by using a lazily generated domain of possible local agents' plans. The DisCSP models the choice of coordination points of each agent considering only sound local plans. If the DisCSP solver finds a solution, the global plan is directly reconstructed from the local plans since the DisCSP solution guarantees compatibility among the underlying local plans.

DPGM, 2010 (implemented in 2013). DPGM [Pellier 2010] also uses a CSP to coordinate the agents' local plans. Unlike Planning First, which uses FF to generate local solutions and delegates the coordination to a DisCSP, the CSP solver in DPGM is explicitly distributed across agents and it is used to extract the local plans from a set of distributed planning graphs. Under the *iterative response planning* strategy introduced by DPGM, the solving process is started by one agent, which proposes a local plan along with a set of coordination constraints. The subsequent agent uses its CSP to extract a local plan compatible with the prior agent's plan and constraints. If an agent is not able to generate a compliant plan, DPGM backtracks to the previous agent, which puts forward an alternative plan with different coordination constraints.

μ-SATPLAN, 2010. μ-SATPLAN [Dimopoulos et al. 2012] is a MAP solver that extends the satisfiability-based planner SATPLAN [Kautz 2006] to a multi-agent context. μ-SATPLAN performs an *a priori* distribution of the MAP task goals, $\mathcal{G}$, among the agents in $\mathcal{AG}$. Similarly to DPGM, agents follow an iterative response planning strategy, where each participant takes the previous agent's solution as an input and extends it to solve its assigned goals via SATPLAN. This way, agents progressively generate a solution.

μ-SATPLAN is limited to collaborative tasks and it is not able to tackle tasks that include cooperative goals because it assumes that each agent can solve its assigned goals by itself. Although μ-SATPLAN was experimentally validated on tasks with only two planning agents, authors claim that the planner is capable of solving tasks with a higher number agents via the application of the iterative response planning scheme.

MAPR, 2013. Similarly to μ-SATPLAN, Multi-Agent Planning by Reuse (MAPR) [Borrajo 2013] allocates the MAP task goals $\mathcal{G}$ among the agents before planning through a relaxed reachability analysis. MAPR grants weak privacy preservation by obfuscating the agents' local tasks. This way, the private information of the local plans is encrypted.

MAPR follows an iterative response planning resolution scheme, like DPGM and μ-SATPLAN. An agent takes the previous agent's solution plan as an input and runs the LAMA planner [Richter and Westphal 2010] to obtain an extended solution plan that attains its allocated agent's goals. The solution plan synthesized by the last agent is a sound global plan for the MAP task, which is then parallelized to ensure that execution agents perform as many actions in parallel as possible. MAPR is limited to collaborative tasks without specialized agents nor cooperative goals. This limitation is a consequence of the assumption that each agent is able to solve its allocated goals by itself, which renders MAPR incomplete.

PMR, 2014. Plan Merging by Reuse (PMR) [Luis and Borrajo 2014; 2015] is a MAP solver that draws upon the goal allocation and obfuscation privacy mechanisms of MAPR. PMR replaces the iterative response planning scheme of MAPR by a post-planning plan merging strategy. Agents carry out the planning stage simultaneously instead of sequentially, and each agent generates local plans for its assigned goals. The resulting local plans are then concatenated, resulting in a sequential global solution. If a solution is not found, the task is solved via a single-agent planner. If the result of the concatenation does not return a sound global plan, agents merge the local plans through a repair procedure. The resulting sequential solution plan is parallelized to ensure the concurrent execution of actions when possible.

CMAP, 2015. CMAP [Borrajo and Fernández 2015] follows the same goal allocation and obfuscation strategy of MAPR and PMR. However, CMAP's resolution scheme merges the encrypted local tasks into a single-agent task ($|\mathcal{AG}| = 1$), which is then solved through the single-agent planner LAMA [Richter and Westphal 2010]. Then, the resulting sequential plan is parallielized to optimize its concurrent execution.

MAP-LAPKT, 2015. Similarly to CMAP, MAP-LAPKT [Muise et al. 2015] conceives a MAP task as a problem that can be solved by a single-agent planner by using the appropriate encoding. More precisely, MAP-LAPKT compiles the MAP task into a single-agent planning task with $|\mathcal{AG}| = 1$ and solves it through the LAPKT toolkit [Ramirez et al. 2015], which provides an independent set of search methods and heuristics. Authors in [Muise et al. 2015] try three different variations of best-first and depth-first search that result in different theoretical properties and performance. The task translation performed by MAP-LAPKT offers weak privacy preservation guarantees.

MARC, 2015. The Multi-Agent Planner for Required Cooperation (MARC) [Sreedharan et al. 2015] is a centralized MAP solver based on the theory of required cooperation [Zhang and Kambhampati 2014]. MARC analyzes the distribution of the MAP task and comes up with a different arrangement of planning agents. More precisely, MARC compiles the original task into a task with a set of *transformer agents*, each one being an ensemble of various agents; i.e., $|\mathcal{AG}_{MARC}| < |\mathcal{AG}|$. A transformer agent comprises the representation of various agents of the original MAP task by using all their actions. The current implementation of MARC merges all the agents in $\mathcal{AG}$ into a single transformer agent ($|\mathcal{AG}_{MARC}| = 1$) to obtain a solution plan; otherwise, the task is solved through a single-agent classical planner.

The resolution scheme of MARC involves three different stages. First, the original MAP task is compiled into a transformer agent task with $|\mathcal{AG}_{MARC}| = 1$. Next, MARC synthesizes a solution for the transformer agent task through the classical planner FD [Helmert 2006] or the portfolio planner IBACOP [Cenamor et al. 2014]. Finally, the solution plan is translated into a solution for the original MAP task, converting each action of the transformer agent plan into multiple agent-specific actions [Sreedharan et al. 2015]. MARC guarantees weak privacy since private elements of the MAP task are occluded in the transformer agent task.

ADP, 2013. The Agent Decomposition-based Planner (ADP) [Crosby et al. 2013] is a factored planning solver that exploits the inherently multi-agent structure (agentization) of some classical STRIPS-style planning tasks and comes up with a MAP task where $|\mathcal{AG}| > 1$. ADP applies a state-based centralized planning procedure to solve the MAP task. In each iteration, ADP determines a set of subgoals that can be attained from the current state by one of the agents. A search process, guided through the well-known h_{FF} heuristic [Hoffmann and Nebel 2001], is then applied to find a plan that achieves those subgoals, thus resulting in a new state. This mechanism iterates successively until a solution is found.

Distoplan, 2010. Distoplan [Fabre et al. 2010] is a factored planning approach that exploits independence within a planning task. Unlike other factored methods [Brafman and Domshlak 2008; Kelareva et al. 2007], Distoplan does not set any bound on the number of actions or coordination points of local plans. In Distoplan, a component or abstraction of the global task is represented as a Finite Automata, which recognizes the regular language formed by the local valid plan of the component. This way, all local plans are manipulated at once and a generic distributed optimization technique, called the message passing algorithm, enables to limit the number of compatible local plans. With this unbounded representation, all valid plans can be computed in one run but stronger conditions are required to guarantee polynomial runtime. Distoplan is the first optimal MAP solver in the literature (note that Planning First is optimal with respect to the number of coordination points, but local planning is carried out through a suboptimal planner).

A#, 2012 (not implemented). In the line of factored planning, A# [Jezequel and Fabre 2012] is a multi-agent A* search that finds a path for the goal in each local component of a task and ensures that the component actions that must be jointly performed are compatible. A# runs modified runs in parallel a modified version of the A* algorithm in each component, and the local search processes are guided towards finding local plans that are compatible with the local plans of the other components. Each local A* finds a plan as a path search in a graph and informs its neighbours of the common actions that may lead to a solution. Particularly, each agent searches its local graph or component while considering the constraints and costs of the rest of agents received through an asynchronous communication mechanism. Authors in [Jezequel and Fabre 2012] do not validate A# experimentally; however, the soundness, completeness and optimality properties of A# are formally proven.

PSM, 2014. PSM [Tožička et al. 2015b; Tožička et al. 2015a] is a recent distributed MAP solver that follows Distoplan's compact representation of local agents plans into Finite Automata that represent sets of local plans, called Planning State Machines (PSMs). Two basic operations are defined over a PSM: obtaining a public projection of the PSM and merging two different PSM. These operations are applied to build a public PSM consisting of merged public parts of individual PSMs. The PSM resolution scheme gradually expands the agents' local PSMs by means of new local plans. A solution for the MAP task is found once the public PSM is not empty. PSM weakly preserves privacy because, despite not revealing private information, it obfuscates states of the PSMs in some situations (see section 3.4.3).

DPP, 2016. The DP-Projection Planner (DPP) [Shani et al. 2016], is a centralized MA-STRIPS solver that uses the Dependency-Preserving (DP) projection, a novel and accurate public projection of the MAP task information. The single planning agent of DPP uses FD [Helmert 2006] to create a high-level plan and then this plan is extended with the agents' private actions via the FF planner [Hoffmann and Nebel 2001], thus resulting in a multi-agent solution plan.

TFPOP, 2011. TFPOP [Kvarnström 2011] is a hybrid approach that combines the flexibility of partial-order planning and the performance of forward-chaining search. In contrast to most MA-STRIPS-based solvers, TFPOP supports temporal reasoning with durative actions. Authors of TFPOP assume that, whereas it is desirable to grant concurrent execution of the executors' plans, an execution agent performs actions sequentially. For this reason, TFPOP synthesizes *threaded partial-order plans*; i.e., non-linear plans that keep a thread of sequentially-ordered actions per agent.

TFPOP follows a centralized approach and synthesizes a solution for multiple executors. The resolution scheme first selects an agent thread to extend and generates

successors by adding one action to the thread and supporting all the preconditions through causal links. Then, TFPOP verifies that the action does not interfere with actions in other threads of the plan, and that there is not any conflict regarding the usage of resources.

4.2. Interleaved Planning and Coordination MAP Solvers

Interleaved MAP solvers apply a form of coordinated search in which agents jointly explore the search space, continually coordinating their activities. Thus, planning and coordination in this setting are inseparable and take place alternatively, in an intertwined manner. The development of interleaved MAP solvers heavily relies on the design of robust communication protocols to coordinate agents during planning.

MAP-POP, FMAP, MH-FMAP, 2010-2015. The leitmotiv of this family of MAP solvers draws upon a distributed exploration of the plan space by multiple agents. Agents locally synthesize plans through an embedded partial-order planning (POP) component and they build a multi-agent search tree by following an A* search strategy which uses global heuristic functions.

MAP-POP [Torreño et al. 2012; 2014a] carries out an incomplete search based on backward POP algorithms and POP heuristics. FMAP [Torreño et al. 2014b] introduces a sound and complete resolution scheme based on forward-chaining POP [Benton et al. 2012]. FMAP search is governed through h_{DTG}, a novel state-based heuristic based on Domain Transition Graphs (DTGs) [Helmert 2006]. Finally, MH-FMAP [Torreño et al. 2015] introduces a multi-heuristic search approach that alternates h_{DTG} and h_{Land}, and builds a Landmark Graph (LG) to estimate the number of pending landmarks of the partial-order plans. The three approaches guarantee weak privacy since private information is occluded throughout the planning process and heuristic evaluation. The h_{Land} estimator resorts to a form of obfuscation during the construction of the LG.

MADLA, 2013. The Multiagent Distributed and Local Asynchronous Planner (MADLA) [Štolba and Komenda 2013; 2014] is a centralized solver that applies global and projected (local) heuristics that require no communication among agents in a distributed state-based search. More precisely, MADLA combines two multi-agent variants of the h_{FF} heuristic [Hoffmann and Nebel 2001], a projected version locally computed by each agent (h_L), and a global version (h_D). MADLA tries to evaluate as many states as possible using the global heuristic h_D, which is more informative than h_L. However, if h_D is busy evaluating another state, the quality of the current state is estimated through h_L. This way, MADLA can use a computationally hard global heuristic without blocking the planning process, thus improving the performance of the system.

MAFS, MAD-A, 2012-2014.* Planning First lacks practical efficiency in large MAP tasks, which motivated its authors to propose a MA-STRIPS approach inspired by efficient classical planners that use state-based heuristics. The Multi-Agent Forward Search (MAFS) [Nissim and Brafman 2014] is a distributed forward-chaining MA-STRIPS-based system in which agents jointly explore the state space. An agent in MAFS maintains an independent open list of states and expands the best one according to its local heuristic estimates. To optimize search, only relevant states are shared; that is, an agent i sends a state S to an agent j if j has at least one action whose public preconditions hold in S, and the action that produces S is public.

Authors also introduce an cost-optimal variation of MAFS, the Multi-Agent Distributed A*, MAD-A* [Nissim and Brafman 2012]. In this case, each agent expands the state that minimizes $f = g + h$, where h is estimated through an admissible heuristic. Particularly, authors tested the landmark heuristic LM-cut [Helmert and Domshlak

2009] and the abstraction heuristic Merge&Shrink [Helmert et al. 2007]. MAD-A* is the first distributed and interleaved solver based on MA-STRIPS.

Secure-MAFS, 2015 (not implemented). Secure-MAFS [Brafman 2015] is an extension of MAFS towards secure MAP, and it is currently the only solver that offers strong privacy guarantees. Authors prove strong privacy for a subset of *logistics* tasks. As opposite to MAFS, an agent i in Secure-MAFS only sends a state S to relevant agents in $\mathcal{AG}$ after verifying that no other state that contains the same non-private information has already been communicated. This way, Secure-MAFS guarantees that agents cannot deduce any private information. Currently, Secure-MAFS is a theoretical work that has not been yet implemented nor experimentally evaluated.

GPPP, 2014. The Greedy Privacy-Preserving Planner (GPPP) [Maliah et al. 2014; 2016] builds upon MAFS and improves its performance via a global landmark-based heuristic function. GPPP applies first a global planning stage and then a local planning stage. In the former, agents agree on a joint coordination scheme by solving a relaxed MAP task that only contains public actions, thereby preserving privacy. As a result, agents obtain a skeleton plan made of public actions only. In the local stage, agents individually calculate private plans to achieve the preconditions of the actions in the skeleton plan. Since the joint coordination is done over a relaxed MAP task, the individual plans of the agents may not succeed at solving the actions preconditions. In this case, the global planning stage is executed again to generate a different coordination scheme, until a solution is found. In GPPP, agents weakly preserve privacy by obfuscating the private information of the shared states through private state identifiers.

MAPlan, 2015. MAPlan [Fišer et al. 2015] is a heuristic MAFS-based solver that adapts several concepts from MAD-A* and MADLA. MAPlan is a distributed and flexible approach that implements a collection of state-space search methods, such as best first or A*, as well as several local and global heuristic functions (h_{FF}, LM-cut, potential heuristics and others), which allows the solver to be run in different configurations. MAPlan can be executed in a single-machine, using local communications, or in a distributed fashion, where each agent is in a different machine and communications are implemented through network message passing. Regarding privacy, MADLA applies a form of obfuscation, replacing private facts in search states by unique local identifiers, which grants weak privacy.

5. ONGOING TRENDS IN DISTRIBUTED AND MULTI-AGENT PLANNING

This section sketches the ongoing and future directions in MAP. We focus on two different topics that have recently captured the attention of the MAP community and constitute the current main research trends in this area; namely, privacy and practical applications.

The state of the art in MAP shows a growing effort in analyzing and formalizing privacy in MAP solvers. On the one hand, the literature includes different approaches to model private information and to define information sharing in privacy-preserving MAP. Moreover, the particular implementation of a MAP solver may jeopardize privacy, since in many cases an agent may be able to infer private information from the received public data. Aside from the four-level classification disclosed in section 3.4.3, there are other recent attempts to theoretically quantify the privacy guarantees of a MAP solver [Štolba et al. 2016c].

On the other hand, some recent approaches to MAP make a smart use of privacy to increase the performance of the solver. A paradigmatic example of this trend is DPP [Shani et al. 2016], which calculates the Dependency-Preserving (DP) projection, an

accurate public projection of the MAP task, which is used to obtain a robust high-level plan that is completed afterwards with private actions. This scheme outperforms a general search and vastly reduces the communication requirements.

MAP has been used in a great variety of practical applications, like in product assembly problems in industry (e.g., car assembly). Agents plan the manufacturing path of the product through the assembly line, which is composed of a number of interconnected resources that can individually perform different operations. ExPlanTech, for instance, is a consolidated framework for agent-based production planning, manufacturing, simulation and supply chain management [Pechoucek et al. 2007].

MAP has also been used to control the flow of electricity in the Smart Grid [Reddy and Veloso 2011]. The agents' actions are individually rational and contribute to desirable global goals such as promoting the use of renewable energy, encouraging energy efficiency and enabling distributed fault tolerance. Another interesting application of MAP is the automated creation of workflows in biological pathways like the Multi-Agent System for Genomic Annotation (BioMAS) [Decker et al. 2002]. This system uses DECAF, a toolkit that provides standard services to integrate agent capabilities, and incorporates a GPGP [Lesser et al. 2004] to coordinate multi-agent tasks.

In decentralized control problems, MAP is applied in coordination of space rovers and helicopter flights, multi-access broadcast channels, and sensor network management, among others [Seuken and Zilberstein 2008]. MAP combined with argumentation techniques to handle belief changes about the context has been used in applications of ambient intelligence in the field of healthcare [Pajares and Onaindia 2013].

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

REFERENCES

J. Benton, Amanda J. Coles, and Andrew I. Coles. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*. 2–10.

Andrea Bonisoli, Alfonso E. Gerevini, Alessandro Saetti, and Ivan Serina. 2014. A Privacy-preserving Model for the Multi-agent Propositional Planning Problem. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*. 973–974.

Daniel Borrajo. 2013. Multi-Agent Planning by Plan Reuse. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*. 1141–1142.

Daniel Borrajo and Susana Fernández. 2015. MAPR and CMAP. In *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*. 1–3.

Ronen I. Brafman. 2015. A Privacy Preserving Algorithm for Multi-Agent Planning and Search. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*. 1530–1536.

Ronen I. Brafman and Carmel Domshlak. 2006. Factored Planning: How, When, and When Not. In *Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference*. 809–814.

Ronen I. Brafman and Carmel Domshlak. 2008. From One to Many: Planning for Loosely Coupled Multi-Agent Systems. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS)*. 28–35.

Isabel Cenamor, Tomás de la Rosa, and Fernando Fernández. 2014. IBACOP and IBACOP2 planner. In *Proceedings of the International Planning Competition (IPC)*.

Matthew Crosby, Michael Rovatsos, and Ronald P.A. Petrick. 2013. Automated Agent Decomposition for Classical Planning. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*. 46–54.

Mathijs de Weerdt and Bradley J. Clement. 2009. Introduction to planning in multiagent systems. *Multiagent and Grid Systems* 5, 4 (2009), 345–355.

Keith Decker, S. Khan, C.J. Schmidt, G. Situ, R. Makkena, and D. Michaud. 2002. BioMAS: a multi-agent system for genomic annotation. *International Journal of Cooperative Information Systems* 11, 3 (2002), 265–292.

Keith Decker and Victor R. Lesser. 1992. Generalizing the Partial Global Planning Algorithm. *International Journal of Cooperative Information Systems* 2, 2 (1992), 319–346.

Marie E. desJardins, Edmund H. Durfee, Charles L. Ortiz, and Michael J. Wolverton. 1999. A survey of research in distributed continual planning. *AI Magazine* 20, 4 (1999), 13–22.

Yannis Dimopoulos, Muhammad A. Hashmi, and Pavlos Moraitis. 2012. μ-SATPLAN: Multi-agent planning as satisfiability. *Knowledge-Based Systems* 29 (2012), 54–62.

Eric Fabre, Loïg Jezequel, Patrik Haslum, and Sylvie Thiébaux. 2010. Cost-Optimal Factored Planning: Promises and Pitfalls. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*. 65–72.

Boi Faltings, Thomas Léauté, and Adrian Petcu. 2008. Privacy guarantees through distributed constraint satisfaction. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, Vol. 2. IEEE, 350–358.

Richard Fikes and Nils J. Nilsson. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2, 3 (1971), 189–208.

Daniel Fišer, Michal Štolba, and Antonín Komenda. 2015. MAPlan. In *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*. 8–10.

Maria Fox, Andrea Gerevini, Derek Long, and Ivan Serina. 2006. Plan stability: replanning versus plan repair. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS06)*. 212–221.

Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela M. Veloso, Daniel Weld, and David Wilkins. 1998. PDDL - The Planning Domain Definition Language. *AIPS-98 Planning Committee* (1998).

Malte Helmert. 2004. A Planning Heuristic Based on Causal Graph Analysis. *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS)* (2004), 161–170.

Malte Helmert. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26, 1 (2006), 191–246.

Malte Helmert and Carmel Domshlak. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway?. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*. 162–169.

Malte Helmert, Patrik Haslum, and Jörg Hoffmann. 2007. Flexible Abstraction Heuristics for Optimal Sequential Planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*. 176–183.

Jörg Hoffmann and Bernhard Nebel. 2001. The FF Planning System: Fast Planning Generation Through Heuristic Search. *Journal of Artificial Intelligence Research* 14 (2001), 253–302.

Loïg Jezequel and Eric Fabre. 2012. A#: A distributed version of A* for factored planning. In *Proceedings of the 51th IEEE Conference on Decision and Control, (CDC)*. 7377–7382.

Henry A. Kautz. 2006. Deconstructing planning as satisfiability. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 21. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 1524.

Elena Kelareva, Olivier Buffet, Jinbo Huang, and Sylvie Thiébaux. 2007. Factored Planning Using Decomposition Trees. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*. 1942–1947.

Daniel L. Kovacs. 2011. *Complete BNF description of PDDL3.1*. Technical Report.

Daniel L. Kovacs. 2012. A Multi-Agent Extension of PDDL3.1. In *Proceedings of the 3rd Workshop on the International Planning Competition (IPC)*. 19–27.

Jonas Kvarnström. 2011. Planning for Loosely Coupled Agents Using Partial Order Forward-Chaining. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI, 138–145.

V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. Prasad, A. Raja, and others. 2004. Evolution of the GPGP/TAEMS domain-independent coordination framework. *Autonomous agents and multi-agent systems* 9, 1-2 (2004), 87–143.

Nerea Luis and Daniel Borrajo. 2014. Plan Merging by Reuse for Multi-Agent Planning. In *Proceedings of the 2nd ICAPS Workshop on Distributed and Multi-Agent Planning (DMAP)*. 38–44.

Nerea Luis and Daniel Borrajo. 2015. PMR: Plan Merging by Reuse. In *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*. 11–13.

Shlomi Maliah, Guy Shani, and Roni Stern. 2014. Privacy Preserving Landmark Detection. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*. 597–602.

Shlomi Maliah, Guy Shani, and Roni Stern. 2015. Privacy Preserving Pattern Databases. *Proceedings of the 3rd ICAPS Workshop on Distributed and Multi-Agent Planning (DMAP)* (2015), 9.

Shlomi Maliah, Guy Shani, and Roni Stern. 2016. Collaborative privacy preserving multi-agent planning. *Autonomous Agents and Multi-Agent Systems* (2016), 1–38. DOI:http://dx.doi.org/10.1007/s10458-016-9333-9

Christian Muise, Nir Lipovetzky, and Miquel Ramirez. 2015. MAP-LAPKT: Omnipotent Multi-Agent Planning via Compilation to Classical Planning. In *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*. 14–16.

Raz Nissim and Ronen I. Brafman. 2012. Multi-agent A* for parallel and distributed systems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 1265–1266.

Raz Nissim and Ronen I. Brafman. 2014. Distributed Heuristic Forward Search for Multi-agent Planning. *Journal of Artificial Intelligence Research* 51 (2014), 293–332.

Raz Nissim, Ronen I. Brafman, and Carmel Domshlak. 2010. A general, fully distributed multi-agent planning algorithm. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 1323–1330.

Sergio Pajares and Eva Onaindia. 2013. Context-aware multi-agent planning in intelligent environments. *Information Sciences* 227 (2013), 22–42.

Michal Pechoucek, Martin Rehák, Petr Charvát, Tomáš Vlcek, and Michal Kolar. 2007. Agent-based approach to mass-oriented production planning: case study. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 37, 3 (2007), 386–395.

Damien Pellier. 2010. Distributed Planning through Graph Merging. In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence (ICAART 2010)*. 128–134. DOI:http://dx.doi.org/10.5220/0002702601280134

Miquel Ramirez, Nir Lipovetzky, and Christian Muise. 2015. Lightweight Automated Planning ToolKiT. http://lapkt.org/. (2015).

Prashant P. Reddy and Manuela M. Veloso. 2011. Strategy learning for autonomous agents in smart grid markets. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*. 1446–1451.

Silvia Richter and Matthias Westphal. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39, 1 (2010), 127–177.

Óscar Sapena, Eva Onaindia, Antonio Garrido, and Marlene Arangú. 2008. A distributed CSP approach for collaborative planning systems. *Engineering Applications of Artificial Intelligence* 21, 5 (2008), 698–709.

Emilio Serrano, Jose M. Such, Juan A. Botía, and Ana García-Fornes. 2013. Strategies for avoiding preference profiling in agent-based e-commerce environments. *Applied Intelligence* (2013), 1–16.

Sven Seuken and Shlomo Zilberstein. 2008. Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems* 17, 2 (2008), 190–250.

Guy Shani, Shlomi Maliah, and Roni Stern. 2016. Stronger Privacy Preserving Projections for Multi-Agent Planning. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS)*. 221–229.

Claude E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27, 3 (1948), 379–423.

Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. 2015. A First Multi-agent Planner for Required Cooperation (MARC). In *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*. 17–20.

Michal Štolba, Daniel Fišer, and Antonín Komenda. 2015. Admissible Landmark Heuristic for Multi-Agent Planning. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS)*. 211–219.

Michal Štolba, Daniel Fišer, and Antonín Komenda. 2016a. Potential Heuristics for Multi-Agent Planning. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS)*. 308–316.

Michal Štolba and Antonín Komenda. 2013. Fast-forward heuristic for multiagent planning. In *Proceedings of the 1st ICAPS Workshop on Distributed and Multi-Agent Planning (DMAP)*. 75–83.

Michal Štolba and Antonín Komenda. 2014. Relaxation Heuristics for Multiagent Planning. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS)*. 298–306.

Michal Štolba and Antonín Komenda. 2015. MADLA: Planning with Distributed and Local Search. In *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*. 21–24.

Michal Štolba, Antonín Komenda, and Daniel L. Kovacs. 2016b. Competition of Distributed and Multiagent Planners (CoDMAP). In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (What's Hot Track)*.

Michal Štolba, Jan Tožička, and Antonín Komenda. 2016c. Quantifying Privacy Leakage in Multi-Agent Planning. *Proceedings of the 4rd ICAPS Workshop on Distributed and Multi-Agent Planning (DMAP)* (2016), 80–88.

Alejandro Torreño, Eva Onaindia, and Óscar Sapena. 2012. An approach to multi-agent planning with incomplete information. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*, Vol. 242. IOS Press, 762–767.

Alejandro Torreño, Eva Onaindia, and Óscar Sapena. 2014a. A Flexible Coupling Approach to Multi-Agent Planning under Incomplete Information. *Knowledge and Information Systems* 38, 1 (2014), 141–178.

Alejandro Torreño, Eva Onaindia, and Óscar Sapena. 2014b. FMAP: Distributed cooperative multi-agent planning. *Applied Intelligence* 41, 2 (2014), 606–626. DOI:http://dx.doi.org/10.1007/s10489-014-0540-2

Alejandro Torreño, Eva Onaindia, and Óscar Sapena. 2015. Global Heuristics for Distributed Cooperative Multi-Agent Planning. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS)*. 225–233.

Alejandro Torreño, Óscar Sapena, and Eva Onaindia. 2015. MH-FMAP: Alternating Global Heuristics in Multi-Agent Planning. In *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*. 25–28.

Jan Tožička, Jan Jakubuv, and Antonín Komenda. 2014. Generating Multi-Agent Plans by Distributed Intersection of Finite State Machines. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*. 1111–1112.

Jan Tožička, Jan Jakubuv, and Antonín Komenda. 2015a. PSM-based Planners Description for CoDMAP 2015 Competition. In *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP-15)*. 29–32.

Jan Tožička, Jan Jakubuv, Antonín Komenda, and Michal Pěchouček. 2015b. Privacy-concerned multiagent planning. *Knowledge and Information Systems* (2015), online pre–print. DOI:http://dx.doi.org/10.1007/s10115-015-0887-7

Roman van der Krogt. 2007. Privacy Loss in Classical Multiagent Planning. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*. 168–174.

Roman van der Krogt. 2009. Quantifying privacy in multiagent planning. *Multiagent and Grid Systems* 5, 4 (2009), 451–469.

Michael Wooldridge. 1997. Agent-Based Software Engineering. *IEE Proceedings - Software Engineering* 144, 1 (1997), 26–37.

Jian F. Zhang, Xuan T. Nguyen, and Ryszard Kowalczyk. 2007. Graph-based Multi-agent Replanning Algorithm. In *Proceedings of the 6th Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 798–805.

Yu Zhang and Subbarao Kambhampati. 2014. A Formal Analysis of Required Cooperation in Multi-agent Planning. *CoRR* abs/1404.5643 (2014). http://arxiv.org/abs/1404.5643

Received July 2016; revised -; accepted -

Online Appendix to: Distributed and Multi-Agent Planning: A Survey

ALEJANDRO TORREÑO, Universitat Politècnica de València
EVA ONAINDIA, Universitat Politècnica de València
ANTONÍN KOMENDA, Czech Technical University in Prague
MICHAL ŠTOLBA, Czech Technical University in Prague

A. RESULTS OF THE 2015 COMPETITION OF DISTRIBUTED AND MULTIAGENT PLANNING

Directly comparing MAP solvers from the taxonomy presented in section 4 is cumbersome because the input format of the planning tasks and output format of the solution plans is not unified. This issue was one for the main motivations for the organization of the first Competition of Distributed and Multiagent Planners[4] (CoDMAP). More than a half of the MAP solvers listed in section 4 entered the competition and were made compatible with the *MA-PDDL* input language and the CoDMAP output format.

A.1. List of Participating Solvers

In order to make the CoDMAP competition as open as possible to various planning paradigms and implementations, it was split into two tracks. The *Centralized Track* served as a transitional track with less strict rules, where planners run on one machine centrally (possibly on more threads in parallel though). The *Distributed Track* forced stricter rules and also required the planners to consume distributed input and run on multiple physical machines in a distributed fashion (each planning agent on one machine). In both tracks, the planning systems were evaluated separately, therefore different planners were not affecting each other. A restricted variant of the Multi-agent Planning Domain Definition Language (*MA-PDDL*) [Kovacs 2012] was used in both tracks and all planners were required by the rules to accept the planning tasks in *MA-PDDL*.

The two Tables below describe the features of the MAP solvers that took part in the Centralized and Distributed Track, respectively. Despite not being mandatory in the Centralized Track, all the participating solvers (with the only exception of ADP) partition the actions among the set of agents described in the *MA-PDDL* task and most of the solvers stick to the *MA-PDDL* definition of private and public facts and actions as well. Since action partitioning follows the *MA-STRIPS* definition, all MAP solvers competing in the Centralized Track are also *MA-STRIPS*-compatible.

In the Distributed Track, it was compulsory to apply both task partitioning and privacy preservation as defined in the factored *MA-PDDL* task descriptions. These three approaches to MAP, together with MAD-A*, are currently the only implementations which provide distributed run. Moreover, MH-FMAP is the only distributed planner able to utilize more than one computational thread for each particular agent.

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

DOI: 0000001.0000001

Table IV. Summary of the MAP solvers competing in the Centralized Track of the 2015 CoDMAP. The coordination division is not strict.

Planner	Coord.	Complete.	Optimal.	*MA-PDDL*		*MA-STRIPS* compatible	Priv.	Single-/multi-thr.
				Factored	Privacy			
ADP	IL	Yes	No	No*	No	No	N	S
CMAP	IL	Yes	No	Yes	Yes	Yes	OB	S
GPPP	IL	Yes	No	Yes	Yes	Yes	OB	S
MADLA	IL	Yes	No	Yes	No^{+}	Yes	OB	M
MAP-LAPKT	IL	Yes	Yes$^{\times}$/No	Yes	Yes	Yes	OB	S
MAPlan	IL	Yes	Yes/No	Yes/No	Yes/No^{+}	Yes	OB	M
MAPR	UT	No	No	Yes	Yes	Yes	OB	S
MARC	IL	Yes	No	Yes	Yes	Yes	OC	S
MH-FMAP	IL	Yes	No	Yes	Yes	Yes	OC	M
PMR	UT	No	No	Yes	Yes	Yes	OB	S
PSM	UT	Yes	No	Yes	No	Yes	OB	M

Coordination: UT - unthreaded, IL - interleaved
Privacy: N - no privacy, OB - obfuscation (weak), OC - occlusion (weak)
Notes: * automated factorization, $^{+}$ by *MA-STRIPS*, $^{\times}$ asymptotically

Table V. Summary of the MAP solvers competing in the Distributed Track (each agent on an independent machine) of the 2015 CoDMAP. The coordination division is not strict. In contrast to the centralized planners, the multi-threaded capability here means that an agent can run in a multi-threaded fashion.

Planner	Coord.	Complete.	Optimal.	*MA-PDDL*		*MA-STRIPS* compatible	Priv.	Single-/multi-thr.
				Factored	Privacy			
MAPlan	IL	Yes	Yes/No	Yes	Yes	Yes	OB	S
MH-FMAP	IL	Yes	No	Yes	Yes	Yes	OC	M
PSM	UT	Yes	No	Yes	Yes	Yes	OB	S

Coordination: UT - unthreaded, IL - interleaved
Privacy: OB - obfuscation (weak), OC - occlusion (weak)

A.2. Summary of the CoDMAP Results

The results of the 2015 CoDMAP are summarized in the Table below. The left-hand side of the Table displays the results of the MAP solvers competing in the Centralized Track, while the right-hand side collects the results of the Distributed Track.

Three of the metrics used in the International Planning Competition[5] (IPC) tracks were used in the CoDMAP to compare the participating MAP solvers:

— Coverage (number of solved tasks) over all domains.
— IPC score over the plan quality Q (a sum over all tasks $\frac{Q^*}{Q}$, where Q^* is the cost of the optimal plan or of the best plan found by any of the participating solvers).
— IPC Agile score over the planning time T (a sum over all the tasks $\frac{1}{(1+\log_{10}(T/T^*))}$, where T^* is the runtime of the fastest planner).

In the Distributed Track, plan quality was evaluated both in terms of total cost, which is the sum of costs of all the actions in a plan, and makespan or duration, defined as the length of the longest agent's plan before reaching a goal. All actions were considered to have a duration of one unit, i.e., non-durative (cf. durative actions in temporal planning).

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

Table VI. Summary of the CoDMAP results. The used metrics are coverage (cvg.) of solved problems from the set of 240 problems in 12 domains, IPC quality score and IPC agile score.

Centralized Track				Distributed Track			
Planner	Cvg.	Quality	Agility	Planner	Cvg.	Quality	Agility
ADP	219	176.23	157.14	PSM	180	140.18	126.75
MAP-LAPKT (t)	217	166.33	165.48	MAPlan	174	134.51	158.83
MAP-LAPKT (q)	217	201.23	71.22	MH-FMAP	107	99.87	52.25
MARC	216	166.20	138.52				
CMAP (t)	211	172.59	161.50				
CMAP (q)	199	188.95	62.51				
MAPlan	197	129.90	114.88				
GPPP	184	149.28	85.43				
PSM	167	111.94	69.44				
MADLA	158	88.31	76.96				
PMR	147	114.68	90.57				
MAPR	139	112.52	106.24				
MH-FMAP	81	80.82	42.36				