Cooperative Multi-Agent Planning: A Survey

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

Cooperative multi-agent planning (MAP) is a relatively recent research field that combines technologies, algorithms, and techniques developed by the Artificial Intelligence Planning and Multi-Agent Systems communities. While planning has been generally treated as a single-agent task, MAP generalizes this concept by considering multiple intelligent *agents* that work cooperatively to develop a course of action that satisfies the goals of the group.

This article reviews the most relevant approaches to MAP, putting the focus on the solvers that took part in the 2015 Competition of Distributed and Multi-Agent Planning, and classifies them according to their key features and relative performance.

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, and Michal Štolba. 2017. Cooperative Multi-Agent Planning: A Survey. *ACM Comput. Surv.* 50, 6, Article 84 (November 2017), 32 pages.
https://doi.org/10.1145/3128584

1 INTRODUCTION

Automated Planning is the field devoted to studying the reasoning side of acting. From the restricted conceptual model assumed in classical planning to the extended models that address temporal planning, online planning or planning in partially observable and non-deterministic domains, the field of Automated Planning has experienced huge advances [36].

Multi-Agent Planning (MAP) introduces a new perspective in the resolution of a planning task with the adoption of a distributed problem-solving scheme instead of the classical single-agent planning paradigm. Distributed planning is required "when planning knowledge or responsibility

This work is supported by the GLASS project Grant No. TIN2014-55637-C2-2-R MINECO of the Spanish Ministerio de Economía, Industria y Competitividad, the Prometeo project II/2013/019 funded by the Valencian Government, and the four-year FPI-UPV research scholarship granted to the first author by the Universitat Politècnica de València. Additionally, this research was partially supported by the Czech Science Foundation under Grant No.15-20433Y CSF.
Authors' addresses: A. Torreño and E. Onaindia, Universitat Politècnica de València, Camino de Vera, s/n, Valencia, 46022, Spain; emails: {atorreno, onaindia}@dsic.upv.es; A. Komenda and M. Štolba, Czech Technical University in Prague, Zikova 1903/4, 166 36, Prague, Czech Republic; emails: antonin.komenda@fel.cvut.cz, michal.stolba@agents.fel.cvut.cz.

is distributed among agents or when the execution capabilities that must be employed to successfully achieve objectives are inherently distributed" [22].

The authors of Reference [22] analyze distributed planning from a twofold perspective; one approach, named *Cooperative Distributed Planning*, regards a MAP task as the process of formulating or executing a plan among a number of participants; the second approach, named *Negotiated Distributed Planning*, puts the focus on coordinating and scheduling the actions of multiple agents in a shared environment. The first approach has evolved to what is nowadays commonly known as *cooperative and distributed MAP*, with a focus on extending planning into a distributed environment and allocating the planning task among multiple agents. The second approach is primarily concerned with controlling and coordinating the actions of multiple agents in a shared environment to ensure that their local objectives are met. We will refer to this second approach, which stresses the coordination and execution of large-scale multi-agent planning problems, as *decentralized planning for multiple agents*. Moreover, while the first planning-oriented view of MAP relies on deterministic approaches, the study of decentralized MAP has yielded an intensive research work on coordination of activities in contexts under uncertainty and/or partial observability with the development of formal methods inspired by the use of Markov Decision Processes [77].

This article surveys deterministic cooperative and distributed MAP methods. Our intention is to provide the reader with a broad picture of the current state of the art in this field, which has recently gained much attention within the planning community thanks to venues such as the Distributed and Multi-Agent Planning workshop[1] and the 2015 Competition of Distributed and Multi-Agent Planning[2] (CoDMAP). Interestingly, although there was a significant amount of work on planning in multi-agent systems in the 1990s, most of this research was basically aimed at developing coordination methods for agents that adopt planning representations and algorithms to carry out their tasks. Back then, little attention was given to the problem of formulating collective plans to solve a planning task. However, the recent CoDMAP initiative of fostering MA-STRIPS—a classical planning model for multi-agent systems [9]—has brought back a renewed interest.

Generally speaking, cooperative MAP is about the collective effort of multiple planning agents to develop solutions to problems that each could not have solved as well (if at all) alone [25]. A cooperative MAP task is thus defined as the collective effort of multiple agents toward achieving a *common goal*, irrespective of how the goals, the knowledge and the agents' abilities are distributed in the application domain. In Reference [18], authors identify several phases to address a MAP task that can be interleaved depending on the characteristics of the problem, the agents and the planning model. Hence, MAP solving may require allocation of goals, formulating plans for solving goals, communicating planning choices and coordinating plans, and execution of plans. The work in Reference [18] is an overview of MAP devoted to agents that plan and interact, presenting a rough outline of techniques for cooperative MAP and decentralized planning. A more recent study examines how to integrate planning algorithms and Belief-Desire-Intention (BDI) agent reasoning [61]. This survey puts the focus on the integration of agent behaviour aimed at carrying out predefined plans that accomplish a goal and agent behaviour aimed at formulating a plan that achieves a goal.

This article presents a thorough analysis of the advances in cooperative and distributed MAP that have lately emerged in the field of Automated Planning. Our aim is to cover the wide and fragmented space of MAP approaches, identifying the main characteristics that define tasks and solvers and establishing a taxonomy of the main approaches of the literature. We explore the great variety of MAP techniques on the basis of different criteria, like agent distribution, communication

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

or privacy models, among others. The survey thus offers a deep analysis of techniques and domain-independent MAP solvers from a broad perspective, without adopting any particular planning paradigm or programming language. Additionally, the contents of this article are geared toward reviewing the broad range of MAP solvers that participated in the 2015 CoDMAP competition.

This survey is structured in five sections. Section 2 offers a historical background on distributed planning with a special emphasis on work that has appeared over the last two decades. Section 3 discusses the main modelling features of a MAP task. Section 4 analyzes the main aspects of MAP solvers, including distribution, coordination, heuristic search, and privacy. Section 5 discusses and classifies the most relevant MAP solvers in the literature. Finally, Section 6 concludes and summarizes the ongoing and future trends and research directions in MAP.

2 RELATED WORK: HISTORICAL BACKGROUND ON MAP

The large body of work on distributed MAP started jointly with an intensive research activity on multi-agent systems (MAS) at the beginning of the 1990s. Motivated by the distributed nature of the problems and reasoning of MAS, decentralized MAP focused on aspects related to distributed control, including activities such as the decomposition and allocation of tasks to agents and utilization of resources [26, 99]; reducing communication costs and constraints among agents [20, 100]; or incorporating group decision making for distributed plan management in collaborative settings (group decisions for selecting a high-level task decomposition or an agent assignation to a task, group processes for plan evaluation and monitoring, etc.) [37]. From this Distributed Artificial Intelligence (DAI) standpoint, MAP is fundamentally regarded as multi-agent *coordination of actions in decentralized systems*.

The inherently distributed nature of tasks and systems also fostered the appearance of techniques for *cooperative formation of global plans*. In DAI, this form of MAP puts greater emphasis on reasoning, stressing the deliberative planning activities of the agents as well as how and when to coordinate such planning activities to come up with a global plan. Given the cooperative nature of the planning task, where all agents are aimed at solving a common goal, this MAP approach features a more centralized view of the planning process. Investigations in this line have yielded a great variety of planning and coordination methods such as techniques to merge the local plans of the agents [13, 21, 27], heuristic techniques for agents to solve their individual sub-plans [28], mechanisms to coordinate concurrent interacting actions [6], or distributed constraint optimization techniques to coordinate conflicts among agents [14]. In this latter work, the authors propose a general framework to coordinate the activities of several agents in a common environment, such as partners in a military coordination, subcontractors working on a building, or airlines operating in an alliance.

Many of the aforementioned techniques and approaches were actually used by some of the early MAP tools. Distributed NOAH [12] is one of the first Partial-Order Planning (POP) systems that generates gradual refinements in the space of (abstract) plans using a representation similar to the Hierarchical Task Networks (HTNs). The scheme proposed in Reference [12] relies on a distributed conflict-solving process across various agents that are able to plan without complete or consistent planning data; the limitation of Distributed NOAH is the amount of information that must be exchanged between planners and the lack of robustness to communication loss or error. In the domain-specific Partial Global Planning (PGP) method [26], agents build their partial global view of the planning problem, and the search algorithm finds local plans that can be then coordinated to meet the goals of all the agents. Generalized PGP (GPGP) is a domain-independent extension of PGP [20, 54] that separates the process of coordination from the local scheduling of activities and task selection, which enables agents to communicate more abstract and hierarchically organized information and has smaller coordination overhead. DSIPE [21] is a distributed version of SIPE-2

[98] closely related to the Distributed NOAH planner. DSIPE proposes an efficient communication scheme among agents by creating partial views of sub-plans. The plan merging process is centralized in one agent and uses the conflict-resolution principle originally proposed in NOAH. The authors of Reference [17] propose a plan merging technique that results in distributed plans in which agents become dependent on each other, but are able to attain their goals more efficiently.

HTN planning has also been exploited for coordinating the plans of multiple agents [11]. The attractiveness of approaches that integrate hierarchical planning in agent teams such as STEAM [87] is that they leverage the abstraction levels of the plan hierarchies for coordinating agents, thus enhancing the efficiency and quality of coordinating the agents' plans. A-SHOP [24] is a multi-agent version of the SHOP HTN planner [63] that implements capabilities for interacting with external agents, performs mixed symbolic/numeric computations, and makes queries to distributed, heterogeneous information sources without requiring knowledge about how and where these resources are located. Moreover, the authors of Reference [48] propose a distributed version of SHOP that runs on a network of clusters through the implementation of a simple distributed backtrack search scheme.

As a whole, cooperative MAP approaches devoted to the construction of a plan that solves a common goal are determined by two factors, the underlying planning paradigm and the mechanism to coordinate the formation of the plan. The vast literature on multi-agent coordination methods is mostly concerned with the task of combining and adapting local planning representations into a global consistent solution. The adaptability of these methods to cooperative MAP is highly dependent on the particular agent distribution and the plan synthesis strategy of the MAP solver. Analyzing these aspects is precisely the aim of the following sections.

3 COOPERATIVE MULTI-AGENT PLANNING TASKS

We define a (cooperative) MAP task as a process in which several agents that are not self-interested work together to synthesize a joint plan that solves a common goal. All agents wish thereby for the goal to be reached at the end of the task execution.

First, this section presents the formalization of the components of a cooperative MAP task. Next, we discuss the main aspects that characterize a MAP task by means of two illustrative examples. Finally, we present how to model a MAP task with *MA-PDDL*, a multi-agent version of the well-known Planning Domain Description Language (*PDDL*) [35].

3.1 Formalization of a MAP Task

Most of the cooperative MAP solvers that will be presented in this survey use a formalism that stems from MA-STRIPS [9] to a lesser or greater extent. For this reason, we will use MA-STRIPS as the baseline model for the formalization of a MAP task. MA-STRIPS is a minimalistic multi-agent extension of the well-known STRIPS planning model [31], which has become the most widely adopted formalism for describing cooperative MAP tasks.

In MA-STRIPS, a MAP task is represented through a finite number of situations or *states*. States are described by a set of *atoms* or *propositions*. States change via the execution of planning *actions*. An action in MA-STRIPS is defined as follows:

Definition 3.1. A **planning action** is a tuple $\alpha = \langle pre(\alpha), add(\alpha), del(\alpha)\rangle$, where $pre(\alpha)$, $add(\alpha)$ and $del(\alpha)$ are sets of atoms that denote the preconditions, *add effects*, and *delete effects* of the action, respectively.

An action α is executable 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 $S' = S \setminus del(\alpha) \cup add(\alpha)$.

Definition 3.2. A **MAP task** is defined as a 5-tuple $\mathcal{T} = \langle \mathcal{AG}, \mathcal{P}, \{\mathcal{A}^i\}_{i=1}^n, \mathcal{I}, \mathcal{G} \rangle$ with the following components:

- $\mathcal{AG}$ is a finite set of n planning entities or agents.
- $\mathcal{P}$ is a finite set of atoms or propositions.
- $\mathcal{A}^i$ is the finite set of planning actions of the agent $i \in AG$. We will denote the set of actions of $\mathcal{T}$ as $\mathcal{A} = \bigcup_{\forall i \in \mathcal{AG}} \mathcal{A}^i$.
- $\mathcal{I} \subseteq \mathcal{P}$ defines the *initial state* of $\mathcal{T}$.
- $\mathcal{G} \subseteq \mathcal{P}$ denotes the *common goal* of $\mathcal{T}$.

A *solution plan* is an ordered set of actions whose application over the initial state $\mathcal{I}$ leads to a state S_g that satisfies the task goals; that is, $\mathcal{G} \subseteq S_g$. In MA-STRIPS a solution plan is defined as a sequence of actions $\Pi_g = \{\Delta, \prec\}$ that attains the task goals, where $\Delta \subseteq \mathcal{A}$ is a non-empty set of actions and $\prec$ is a total-order relationship among the actions of Δ. However, other MAP models assume a more general definition of a plan; for example, as a set of of sequences of actions (one sequence per agent), as in Reference [53]; or as a partial-order plan [88]. In the following, we will consider a solution plan as a set of partially ordered actions.

The action distribution model of MA-STRIPS, introduced in Definition 3.2, classifies each atom $p \in \mathcal{P}$ as either *internal* (private) to an agent $i \in \mathcal{AG}$, if it is only used and affected by the actions in $\mathcal{A}^i$, or *public* to all the agents in $\mathcal{AG}$. $\mathcal{P}^i_{int}$ denotes the atoms that are internal to agent i, while $\mathcal{P}_{pub}$ refers to the public atoms of the task. The distribution of the information of a MAP task $\mathcal{T}$ configures the *local view* that an agent i has over $\mathcal{T}$, $\mathcal{T}^i$, which is formally defined as follows:

Definition 3.3. The **local view** of a task $\mathcal{T} = \langle \mathcal{AG}, \mathcal{P}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$ by an agent $i \in \mathcal{AG}$ is defined as $\mathcal{T}^i = \langle \mathcal{P}^i, \mathcal{A}^i, \mathcal{I}^i, \mathcal{G} \rangle$, which includes the following elements:

- $\mathcal{P}^i = \mathcal{P}^i_{int} \cup \mathcal{P}_{pub}$ denotes the atoms accessible by agent i.
- $\mathcal{A}^i \subseteq \mathcal{A}$ is the set of planning actions of i.
- $\mathcal{I}^i \subseteq \mathcal{P}^i$ is the set of atoms of the initial state accessible by agent i.
- $\mathcal{G}$ denotes the common goal of the task $\mathcal{T}$. An agent i knows all the atoms of $\mathcal{G}$ and it will contribute to their achievement either directly (achieving a goal $g \in \mathcal{G}$) or indirectly (reaching effects that help other agents achieve g).

Note that Definition 3.3 does not specify $\mathcal{G}^i$, a set of individual goals of an agent i, because in a cooperative MAP context the common goal $\mathcal{G}$ is shared among all agents and it is never assigned to some particular agent (see next section for more details).

3.2 Characterization of a MAP Task

This section introduces a brief example based on a logistics domain [90] to illustrate the characteristics of a MAP task.

Consider the transportation task $\mathcal{T}_1$ 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 transport 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 must be collected from area $ga1$. In this task, agents $ta1$ and $ta2$ have the same planning capabilities, but they operate in different geographical areas; that is, 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. For solving this task, $ta1$ will use its truck $t1$ to load the package of raw materials p, initially located in $l2$, and then it will transport

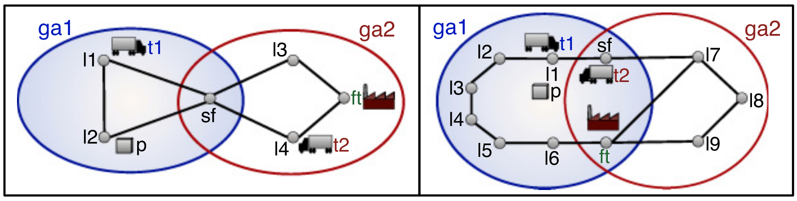

Fig. 1. MAP task examples: task $\mathcal{T}_1$ (left) and task $\mathcal{T}_2$ (right).

Table 1. Task View $\mathcal{T}^i$ for Each Agent i in Example Tasks $\mathcal{T}_1$ and $\mathcal{T}_2$

Task	$\mathcal{T}_1$			$\mathcal{T}_2$		
$\mathcal{AG}$	$ta1$	$ta2$	ft	$ta1$	$ta2$	ft
$\mathcal{P}^i$	(*pos t1* $*$)	(*pos t2* $*$)	(*pending fp*)	(*pos t1* $*$)	(*pos t2* $*$)	(*pending fp*)
	(*at p* $*$)		(*at p ft*)	(*at p* $*$)		(*at p ft*)
	(*manufactured fp*)			(*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*)	(*pos t1 l1*) (*at p l1*) (*at p3 ft*)	(*pos t2 sf*)	
$\mathcal{G}$	(*manufactured fp*)			(*at p ft*)		

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 the factory ft, which will in turn manufacture the final product fp. Therefore, this task involves three specialized agents, which are spatially or functionally distributed, and must cooperate to accomplish the common goal of having a final manufactured product fp.

Task $\mathcal{T}_1$ defines a *group goal*; that is, a goal that requires the participation of all the agents to solve it. Given a task $\mathcal{T}_g = \langle \mathcal{AG}, \mathcal{P}, \mathcal{A}, \mathcal{I}, \{g\} \rangle$, which includes a single goal g, we say that g is a group goal if for every solution plan $\Pi_g = \{\Delta, <\}$, $\exists \alpha, \beta \in \Delta : \alpha \in \mathcal{A}^i, \beta \in \mathcal{A}^j$ and $i \neq j$. We can thus distinguish between group goals, which require the participation of more than one agent, and non-group goals, which can be independently achieved by a single agent.

The presence or absence of group goals often determines the complexity of a cooperative MAP task. Figure 1 (right) depicts task $\mathcal{T}_2$, where the goal is to deliver the package p into the factory ft. This is a non-group goal, because agent $ta1$ is capable of attaining it by itself, gathering p in $l1$ and transporting it to ft through the locations of its area $ga1$. However, the optimal solution for this task is that agent $ta1$ takes p to sf so agent $ta2$ loads then p in sf and completes the delivery to the factory ft through location $l7$. These two examples show that cooperative MAP involves multiple agents working together to solve tasks that they are unable to attain by themselves (task $\mathcal{T}_1$), or tasks that are accomplished better by cooperating (task $\mathcal{T}_2$) [18].

Tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ emphasize most of the key elements of a MAP context. The spatial and/or functional distribution of the participants 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 1. Atoms of the form (*pos t1* $*$) (note that $*$ acts as a wildcard) are accessible to agent $ta1$, since they model the position of truck $t1$. Atoms of the form (*pos t2* $*$), which describe the location of truck $t2$, are accessible to agent $ta2$. Finally, (*pending fp*) belongs to agent ft and denotes that the manufacturing of fp (the goal of task $\mathcal{T}_1$) is still pending.

The atoms related to the location of the product p, (*at p* $*$), as well as (*manufactured fp*), which indicates that the final product fp is already manufactured, are accessible to the three agents, $ta1$, $ta2$ and ft. Since agents ignore the configuration of the working area of the other agents, the knowledge of agent $ta1$ regarding the location of p is restricted to the atoms (*at p l1*), (*at p l2*), (*at p sf*) and (*at p t1*), while agent $ta2$ knows (*at p sf*), (*at p l3*), (*at p l4*), (*at p t2*) and (*at p ft*). The awareness of agent ft with respect to the location of p is limited to (*at p ft*).

The information distribution of a MAP task stresses the issue of *privacy*, which is one of the basic aspects that must be considered in multi-agent applications [76]. Agents manage information that is not relevant for their counterparts or sensitive data of their internal operational mechanisms that they are not willing to disclose. For instance, $ta1$ and $ta2$ cooperate in solving tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ but they could also be potential competitors, since they work in the same business sector. For these reasons, providing privacy mechanisms to guarantee that agents do not reveal the internal configuration of their working areas to each other is a key issue.

In general, agents in MAP seek to minimize the information they share with each other, thus exchanging only the information that is relevant for other participating agents to solve the MAP task.

3.3 Modelling of a MAP Task with MA-PDDL

The adoption of a common language for modelling planning domains allows for a direct comparison of different approaches and increases the availability of shared planning resources, thus facilitating the scientific development of the field [34]. Modelling a cooperative MAP task involves defining several elements that are not present in single-agent planning tasks. Widely adopted single-agent planning task specification languages, such as *PDDL* [35], lack the required machinery to specify a MAP task. Recently, *MA-PDDL*[3], the multi-agent version of *PDDL*, was developed in the context of the 2015 CoDMAP competition [51] as the first attempt to create a *de facto* standard specification language for MAP tasks. We will use *MA-PDDL* as the language for modelling MAP tasks.

MAP solvers that accept an *unfactored* specification of a MAP task use a single input that describes the complete task $\mathcal{T}$. In contrast, other MAP approaches require a *factored* specification; that is, the local view of each agent, $\mathcal{T}^i$. Additionally, modelling a MAP task may require the specification of the private information that an agent cannot share with other agents.

MA-PDDL allows for the definition of both factored (`:factored-privacy` requirement) and unfactored (`:unfactored-privacy` requirement) task representations. To model the transportation task $\mathcal{T}_1$ (see Figure 1 (left) in Section 3.2), we will use the factored specification. Task $\mathcal{T}_1^i$ of agent i is encoded by means of two independent files: the *domain* file describes general aspects of the task ($\mathcal{P}^i$ and $\mathcal{A}^i$, which can be reused for solving other tasks of the same typology); the *problem* file contains a description of the particular aspects of the task to solve ($\mathcal{I}^i$ and $\mathcal{G}$). For the sake of simplicity, we only display fragments of the task $\mathcal{T}_1^{ta1}$.

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

The domain of `transport-agencies` starts with the type hierarchy, which includes the types `transport-agency` and `factory` to define the agents of the task. Note that the type `factory` is defined as a subtype of `place`, because a `factory` is also interpreted as a `place` reachable by a

[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*.

```
(define (domain transport-agency)
  (:requirements :factored-privacy :typing :equality :fluents)
  (:types transport-agency area location package product - object
          truck place - location
          factory - place)
  (:predicates
    (manufactured ?p - product) (at ?p - package ?l - location)
    (:private
      (area ?ag - transport-agency ?a - area) (in-area ?p - place ?a - area)
      (owner ?a - transport-agency ?t - truck) (pos ?t - truck ?l - location)
      (link ?p1 - place ?p2 - place)
    )
  )
  (:action drive
    :parameters   (?ag - transport-agency ?a - area ?t - truck ?p1 - place ?p2 - place)
    :precondition (and (area ?ag ?a) (in-area ?p1 ?a) (in-area ?p2 ?a)
                  (owner ?a ?t) (pos ?t ?p1) (link ?p1 ?p2))
    :effect       (and (not (pos ?t ?p1)) (pos ?t ?p2))
  )
[...]
)
```

Listing 1. Excerpt of the domain file for transport-agency agents

truck. The remaining elements of the task are identified by means of the types `location`, `package`, `truck`, and so on.

The `:predicates` section includes the set of first-order *predicates*, which are patterns to generate the agent's propositions, $\mathcal{P}^i$, through the instantiation of their parameters. The domain for `transport-agencies` includes the public predicates `at`, which models the position of the `packages`, and `manufactured`, which indicates that the task goal of manufacturing a `product` is fulfilled. Despite the fact that only the `factory` agent ft has the ability to manufacture `products`, all the agents have access to the predicate `manufactured` so the `transport-agencies` will be informed when the task goal is achieved.

The remaining predicates in Listing 1 are in the section `:private` of each `transport-agency`, meaning that agents will not disclose information concerning the topology of their working `areas` or the status of their `trucks`.

Finally, the `:action` block of Listing 1 shows the *action schema* `drive`. An action schema represents a number of different actions that can be derived by instantiating its variables. Agents $ta1$ and $ta2$ have three action schemas: `load`, `unload` and `drive`.

Regarding the *problem* description, the factored specification includes three problem files, one per agent, that contain $\mathcal{I}^i$, the initial state of each agent, and the task goals $\mathcal{G}$.

```
(define (problem ta1)
  (:domain transport-agency)
  (:objects
    ta1 - transport-agency
    ga1 - area                  l1 l2 sf - place
    p - package                 fp - product
    (:private t1 - truck)
  )
  (:init
    (area ta1 ga1) (pos t1 l1) (owner t1 ta1) (at p l1)
    (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 (manufactured fp))
)
```

Listing 2. Problem file of agent *ta*1

Listing 2 depicts the problem description of task $\mathcal{T}_1^{ta1}$. The information managed by $ta1$ is related to its `truck t1` (which is defined as `:private` to prevent $ta1$ from disclosing the location or cargo of `t1`), along with the `places` within its working `area`, `ga1`, and the `package p`. The `:init` section of Listing 2 specifies $\mathcal{I}^{ta1}$; that is, the location of `truck t1` and the position of `package p`, which is initially located in `ga1`. Additionally, $ta1$ is aware of the `links` and `places` within its `area` `ga1`. The `:goal` section is common to the three agents in $\mathcal{T}_1$ and includes a single goal indicating that the `product fp` must be `manufactured`.

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. These functionalities make *MA-PDDL* a fairly expressive language to specify MAP tasks.

4 MAIN ASPECTS OF A MAP SOLVER

Solving a cooperative MAP task requires various features such as information distribution, specialized agents, coordination, or privacy. The different MAP solving techniques in the literature can be classified according to the mechanisms they use to address these functionalities. We identify six main features to categorize cooperative MAP solvers:

- **Agent distribution**: From a conceptual point of view, MAP is regarded as a task in which multiple agents are involved, either as entities participating in the plan synthesis (planning agents) or as the target entities of the planning process (actuators or execution agents).
- **Computational process**: From a computational perspective, MAP solvers use a *centralized* or *monolithic* design that solves the MAP task through a central process, or a *distributed* approach that splits the planning activity among several processing units.
- **Plan synthesis schemes**: There exist a great variety of strategies to tackle the process of synthesizing a plan for the MAP task, mostly characterized by how and when the *coordination* activity is applied. Coordination comprises the distributed information exchange processes by which the participating agents organize and harmonize their activities to work together properly.
- **Communication mechanisms**: Communication among agents is an essential aspect that distinguishes MAP from single-agent planning. The type of communication enabled in MAP solvers is highly dependent on the type of computational process (centralized or distributed) of the solver. Thus, we will classify solvers according to the use of internal or external communication infrastructures.
- **Heuristic search**: As in single agent planning, MAP solvers commonly apply heuristic search to guide the planning process. In MAP, we can distinguish between *local* heuristics (each agent i calculates an estimate to reach the task goals $\mathcal{G}$ using only its accessible information in $\mathcal{T}^i$) or *global* heuristics (the estimate to reach $\mathcal{G}$ is calculated among all the agents in $\mathcal{AG}$).
- **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 former MAP solvers [97], 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.

Table 2. Conceptual Schemes According to the Number of Planning and Execution Agents (Along with Example MAP Solvers That Apply Them)

		Planning agents $|\mathcal{AG}|$	
		1	n
Execution agents	1	Single-agent planning FD [39]	Factored planning ADP [16]
	n	Planning *for* multiple agents TFPOP [53]	Planning *by* multiple agents FMAP [91]

4.1 Agent Distribution

Agents in MAP can adopt different roles: planning agents are *reasoning* entities that synthesize the course of action or plan that will be later executed by a set of *actuators* or *execution agents*. An execution agent can be, among others, a robot in a multi-robot system, or a software entity in an execution simulator.

From a conceptual point of view, MAP solvers are characterized by the agent distribution they apply; that is, the number of planning and execution agents involved in the task. Typically, it is assumed that one planning agent from the set $\mathcal{AG}$ of a MAP task $\mathcal{T}$ is associated with one actuator in charge of executing the actions of this planning agent in the solution plan. However, some MAP solvers alter this balance between planning and acting agents.

Table 2 summarizes the different schemes according to the relation between the number of planning and execution agents. Single-agent planning is the simplest mapping: the task is solved by a single planning agent, that is, $|\mathcal{AG}| = 1$, and executed by a single actuator. We can mention Fast Downward (FD) [39] as one of the most utilized single-agent planners within the planning community.

MAP solvers like Distoplan [29], A# [45], and ADP [16] follow a *factored* agent distribution inspired by the factored planning scheme [1]. Under this paradigm, a single-agent planning task is decomposed into a set of independent factors (agents), thus giving rise to a MAP task with $|\mathcal{AG}| > 1$. Then, factored methods to solve the agents' local tasks $\mathcal{T}^i$ are applied, and finally, the computed local plans are pieced together into a valid solution plan [8]. Factored planning exploits locality of the solutions and a limited information propagation between components.

The second row of Table 2 outlines the classification of MAP approaches that build a plan that is conceived to be then executed by several actuators. Some solvers in the literature regard MAP as a single planning agent working *for* a set of actuators ($|\mathcal{AG}| = 1$), while other approaches regard MAP as planning *by* multiple planners ($|\mathcal{AG}| > 1$).

4.1.1 *Planning for Multiple Agents.*

Under this scheme, the actions of the solution plan are distributed among actuators typically via the introduction of constraints. TFPOP [53] applies single-agent planning for multi-agent domains where each execution agent is associated with a sequential thread of actions within a partial-order plan. The combination of forward-chaining and least-commitment of TFPOP provides flexible schedules for the acting agents, which execute their actions in parallel. The work in Reference [15] transforms a MAP task that involves multiple agents acting concurrently and cooperatively in a single-agent planning task. The transformation compels agents to select joint actions associated with a single subset of objects at a time, and ensures that the concurrency constraints on this subset are satisfied.The result is a single-agent planning problem in which agents perform actions individually, one at a time.

The main limitation of this planning-for-multiple-agents scheme is its lack of privacy, since the planning entity has complete access to the MAP task $\mathcal{T}$. This is rather unrealistic if the agents

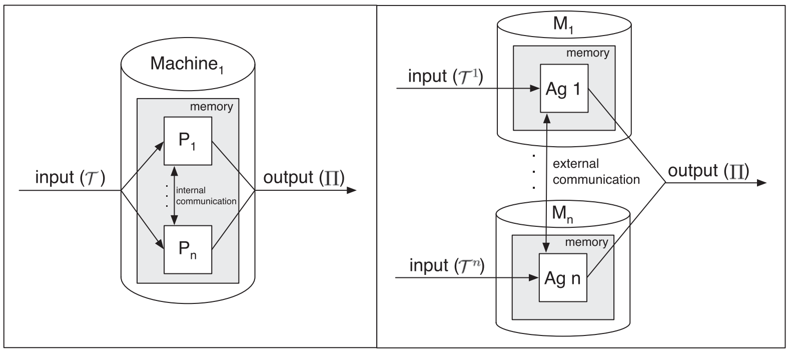

Fig. 2. Centralized (monolithic) vs. distributed (agent-based) implementation.

involved in the task have sensitive private information they are not willing to disclose [75]. Therefore, this scheme is not a suitable solution for privacy-preserving MAP tasks like task $\mathcal{T}_1$ described in Section 3.2.

4.1.2 Planning by Multiple Agents. This scheme distributes the MAP task among several planning agents where each is associated with a local task $\mathcal{T}^i$. Thus, planning-by-multiple-agents puts the focus on the coordination of the planning activities of the agents. Unlike single-planner approaches, the planning decentralization inherent to this scheme makes it possible to effectively preserve the agents' privacy.

In general, solvers that follow this scheme, such as FMAP [88], maintain a one-to-one correspondence between planning and execution agents; that is, planning agents are assumed to solve their tasks, which will be later executed by their corresponding actuators. There exist, however, some exceptions in the literature that break this one-to-one correspondence such as MARC [81], which rearranges the n planning agents in $\mathcal{AG}$ into m *transformer agents* ($m < n$), where a transformer agent comprises the planning tasks of several agents in $\mathcal{AG}$. All in all, MARC considers m reasoning entities that plan for n actuators, where $m < n$.

4.2 Computational Process

From a computational standpoint, MAP solvers are classified as *centralized* or *distributed*. Centralized solvers draw upon a monolithic implementation in which a central process synthesizes a global solution plan for the MAP task. In contrast, distributed MAP methods are implemented as multi-agent systems in which the problem-solving activity is fully decentralized.

Centralized MAP. In this approach, the MAP task $\mathcal{T}$ is solved on a single machine regardless the number of planning agents conceptually considered by the solver. The main characteristic of a centralized MAP approach is that tasks are solved in a monolithic fashion, so all the processes of the MAP solver, $\{P_1, \ldots, P_n\}$, are run in one same machine (see Figure 2 (left)).

The motivation for choosing a centralized MAP scheme is twofold: (1) external communication mechanisms to coordinate the planning agents are not needed; and (2) robust and efficient single-agent planning technology can be easily reused.

Regarding agent distribution, MAP solvers that use a single planning agent generally apply a centralized computational scheme, as for example TFPOP [53] (see Table 2). On the other hand, some algorithms that conceptually rely on the distribution of the MAP task among several planning agents do not actually implement them as software agents, but as a centralized procedure. For example, MAPR [4] establishes a sequential order among the planning agents and applies a centralized planning process that incrementally synthesizes a solution plan by solving the agents' local tasks in the predefined order.

Distributed MAP. Many approaches that conceive MAP as planning by multiple agents (see Table 2), are developed as multi-agent systems (MAS) defined by several independent *software agents*. By software agent, we refer to a computer 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 with other agents via some communication language to achieve its objectives (social ability) [101].

In this context, a software agent of a MAS plays the role of a planning agent in $\mathcal{AG}$. This way, in approaches that follow the *planning by multiple agents* scheme introduced in the previous section, a software agent encapsulates the local task $\mathcal{T}^i$ of a planning agent $i \in \mathcal{AG}$. Given a task, where $|\mathcal{AG}| = n$, distributed MAP solvers can be run on up to n different hosts or machines (see Figure 2 (right)).

The emphasis of the distributed or agent-based computation lies in the coordination of the concurrent activities of the software planning agents. Since agents may be run on different hosts (see Figure 2 (right)), having a proper communication infrastructure and message-passing protocols is vital for the synchronization of the agents.

Distributed solvers like FMAP [90] launch $|\mathcal{AG}|$ software agents that seamlessly operate on different machines. FMAP builds upon the MAS platform Magentix2 [86], which provides the messaging infrastructure for agents to communicate over the network.

4.3 Plan Synthesis Schemes

In most MAP tasks, there are dependencies between agents' actions and none of the participants has sufficient competence, resources or information to solve the entire problem. For this reason, agents must coordinate with each other to cooperate and solve the MAP task properly.

Coordination is a multi-agent process that harmonizes the agents' activities, allowing them to work together in an organized way. In general, coordination involves a large variety of activities, such as distributing the task goals among the agents, making joint decisions concerning the search for a solution plan, or combining the agents' individual plans into a solution for a MAP task. Since coordination is an inherently distributed mechanism, it is only required in MAP solvers that conceptually draw upon multiple planning agents (see right column of Table 2).

The characteristics of a MAP task often determine the coordination requirements for solving the task. For instance, tasks that feature group goals, like the task $\mathcal{T}_1$ depicted in Section 3.2, usually demand a stronger coordination effort. Therefore, the capability and efficiency of a MAP solver is determined by the coordination strategy that governs its behaviour.

The following subsections analyse the two principal coordination strategies in MAP; namely, *unthreaded* and *interleaved* planning and coordination.

4.3.1 Unthreaded Planning and Coordination.

4.3.1 Unthreaded Planning and Coordination. This strategy defines planning and coordination as *sequential* activities, such that they are viewed as two separate black boxes. Under this strategy, an agent $i \in \mathcal{AG}$ synthesizes a plan to its local view of the task, $\mathcal{T}^i$, and coordination takes place *before* or/and *after* planning.

Pre-planning Coordination. Under this plan synthesis scheme, the MAP solver defines the necessary constrains to guarantee that the plans that solve the local tasks of the agents are properly combined into a consistent solution plan for the whole task $\mathcal{T}$ (see Figure 3(a)).

ADP [16] follows this scheme by applying an *agentification* procedure that distributes a STRIPS planning task among several planning agents (see Table 2). More precisely, ADP is a fully auto-

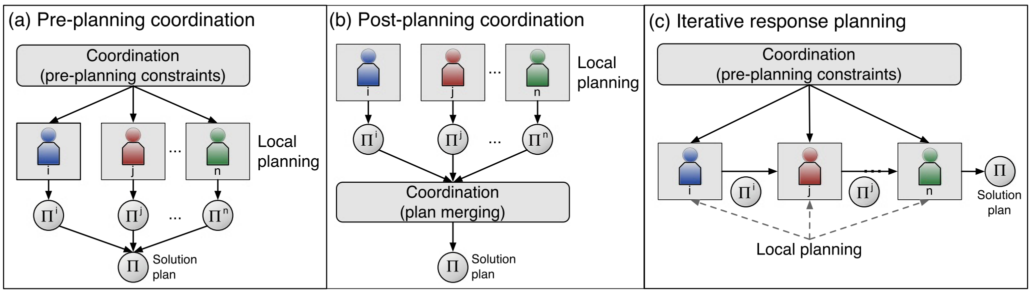
(a) Pre-planning coordination
Coordination (pre-planning constraints)
Local planning
Solution plan
(b) Post-planning coordination
Local planning
Coordination (plan merging)
Solution plan
(c) Iterative response planning
Coordination (pre-planning constraints)
Solution plan
Local planning

Fig. 3. Plan synthesis schemes in unthreaded planning and coordination.

mated 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. By leveraging this agent decomposition, ADP applies a centralized, sequential and total-order planning algorithm that yields a solution for the original STRIPS task. Since the task $\mathcal{T}$ is broken down into several local tasks independent from each other, the local solution plans are consistently combined into a solution for $\mathcal{T}$.

Ultimately, the purpose of pre-planning coordination is to guarantee that the agents' local plans are seamlessly combined into a solution plan that attains the goals of the MAP task, thus avoiding the use of plan merging techniques at post-planning time.

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

In PMR (Plan Merger by Reuse) [56], the local plans of the agents are concatenated into a solution plan for the MAP task. Other post-planning coordination approaches apply an information exchange between agents to come up with the global solution. For instance, PSM [94] draws upon a set of finite automata, called Planning State Machines (PSM), where each automaton represents the set of local plans of a given agent. In one iteration of the PSM procedure, each agent i generates a plan that solves its local task $\mathcal{T}^i$, incorporating this plan into its associated PSM. Then, agents exchange the public projection of their PSMs, until a solution plan for the task $\mathcal{T}$ is found.

Iterative Response Planning. This plan synthesis scheme, firstly introduced by DPGM [70], successively applies a planning-coordination sequence, each corresponding to a planning agent. An agent i receives the local plan of the preceding agent along with a set of constraints for coordination purposes, and *responds* by building up a solution for its local task $\mathcal{T}^i$ on top of the received plan. Hence, the solution plan is incrementally synthesized (see Figure 3(c)).

Multi-Agent Planning by Reuse (MAPR) [4] is an iterative response solver based on *goal allocation*. The task goals $\mathcal{G}$ are distributed among the agents before planning, such that an agent i is assigned a subset $\mathcal{G}^i \subset \mathcal{G}$, where $\bigcap_{i\in\mathcal{AG}} \mathcal{G}^i = \emptyset$. Additionally, agents are automatically arranged in a sequence that defines the order in which the iterative response scheme must be carried out.

In unthreaded planning and coordination schemes, agents do not need communication skills, because they do not interact with each other during planning. This is the reason why the unthreaded strategy is particularly efficient for solving tasks that do not require a high coordination effort. In contrast, it presents several limitations when solving tasks with group goals, due to the fact that agents are unable to discover and address the cooperation demands of other agents. The needs of cooperation that arise when solving group goals are hard to discover at pre-planning time, and

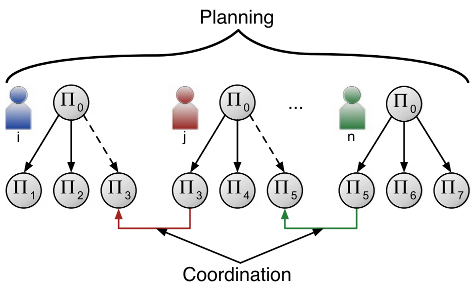

Fig. 4. Multi-agent search in interleaved planning and coordination.

plan merging techniques are designed only to fix inconsistencies among local plans, rather than repairing the plans to satisfy the inter-agent coordination needs. Consequently, unthreaded approaches are more suitable for solving MAP task that do not contain group goals; that is, every task goal can be solved by at least one single agent.

4.3.2 Interleaved Planning and Coordination. A broad range of MAP techniques *interleave* the planning and coordination activities. This coordination strategy is particularly appropriate for tasks that feature group goals, since agents explore the search space jointly to find a solution plan, rather than obtaining local solutions individually. In this context, agents continuously coordinate with each other to communicate their findings, thus effectively intertwining planning and coordination.

Most interleaved solvers, such as MAFS [64] and FMAP [90], commonly rely on a *coordinated multi-agent search* scheme, wherein nodes of the search space are contributed by several agents (see Figure 4). This scheme involves selecting a node for expansion (planning) and exchanging the successor nodes among the agents (coordination). Agents thus jointly explore the search space until a solution is found, alternating between phases of planning and coordination.

Different forms of coordination are applicable in the interleaved resolution strategy. In FMAP [90], agents share an open list of plans and jointly select the most promising plan according to global heuristic estimates. Each agent i expands the selected node using its actions $\mathcal{A}^i$, and then, agents evaluate and exchange all the successor plans. In MAFS [64], each agent i keeps an independent open list of states. Agents carry out the search simultaneously: an agent i selects a state S to expand from its open list according to a local heuristic estimate, and synthesizes all the nodes that can be generated through the application of the actions $\mathcal{A}^i$ over S. Out of all the successor nodes, agents only share the states that are relevant to other agents.

Interleaving planning and coordination is very suitable for solving complex tasks that involve group 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, the interleaved scheme allows agents to efficiently address group goals.

The main drawback of this coordination strategy is the high communication cost in a distributed MAP setting, because alternating planning and coordination usually entails exchanging a high number of messages to continuously coordinate agents.

4.4 Communication Mechanisms

Communication among agents plays a central role in MAP solvers that conceptually define multiple planners (see Table 2), since planning agents must coordinate their activities to accomplish the task goals. As shown in Figure 2, different agent communication mechanisms can be applied, depending on the computational process followed by the MAP solver.

Internal Communication. Solvers that draw upon a centralized implementation resort to *internal* or simulated multi-agent communication. For example, the centralized solver MAPR [4] distributes the task goals, $\mathcal{G}$, among the planning agents, and agents solve their local tasks sequentially. Once a local plan Π^i for an agent $i \in AG$ is computed, the information of Π^i is used as an input for the next agent in the sequence, j, thus simulating a message passing between agents i and j. This type of simple and simulated communication system is all that is required in centralized solvers that run all planning agents in a single machine.

External Communication. As displayed in Figure 2 (right), distributed MAP solvers draw upon *external* communication mechanisms by which different processes (agents), potentially allocated on different machines, exchange messages to interact with each other. External communication can be easily enabled by linking the different agents via network sockets or a messaging broker. For example, agents in MAPlan [32] exchange data over the TCP/IP protocol when the solver is executed in a distributed manner. A common alternative to implement external communication in MAS implies using a message passing protocol compliant with the IEEE FIPA standards [33], which are intended to promote the interoperation of heterogeneous agents. The Magentix2 MAS platform [86] used by MH-FMAP [91] facilitates the implementation of agents with FIPA-compliant messaging capabilities.

The use of external communication mechanisms allows distributed solvers to run the planning agents in decentralized machines and to coordinate their activities by exchanging messages through the network. The flexibility provided by external communication mechanisms comes at the cost of performance degradation. The results of the 2015 CoDMAP competition [51] show that centralized solvers like ADP [16] outperform solvers executed in a distributed setting, such as PSM [93]. Likewise, an analysis performed in Reference [90] reveals that communication among agents is the most time-consuming activity of the distributed approach FMAP, thus compromising the overall scalability of this solver. Nevertheless, the participants in the distributed track of the CoDMAP exhibited a competitive performance, which proves that the development of fully distributed MAP solvers is worth the overhead caused by external communication infrastructures.

4.5 Heuristic Search

Ever since the introduction of HSP [55], the use of heuristic functions that guide the search by estimating the quality of the nodes of the search tree has proven to be one of the most robust and reliable problem-solving strategies in single-agent planning. Over the years, many solvers based on heuristic search, such as FF [42] or LAMA [73], have consistently dominated the International Planning Competitions.[4]

Since most MAP solvers stem from single-agent planning techniques, heuristic search is one of the most common approaches in the MAP literature. In general, in solvers that synthesize a plan for multiple executors, the single planning agent (see Table 2) has complete access to the MAP task $\mathcal{T}$, and so it can compute heuristics that leverage the *global* information of $\mathcal{T}$, in a way that is very similar to that of single-agent planners. However, in solvers that feature multiple planning agents, that is, $|\mathcal{AG}| > 1$, each agent i is only aware of the information defined in $\mathcal{T}^i$ and no agent has access to the complete task $\mathcal{T}$. Under a scheme of planning by multiple agents, one can distinguish between *local* and *global* heuristics.

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

In local heuristics, an agent i estimates the cost of the task goals, $\mathcal{G}$, using only the information in $\mathcal{T}^i$. The simplicity of local heuristics, which do not require any interactions among agents, contrasts with the low accuracy of the estimates they yield due to the limited task view of the agents. Consider, for instance, the example task $\mathcal{T}_1$ presented in Section 3.2: agent $ta1$ does not have sufficient information to compute an accurate estimate of the cost to reach the goals of $\mathcal{T}_1$, since $\mathcal{T}_1^{ta1}$ does not include the configuration of the area $ga2$. In general, if $\mathcal{T}^i$ is a limited view of $\mathcal{T}$, then local heuristics will not yield informative estimates of the cost of reaching $\mathcal{G}$.

In contrast, a global heuristic in MAP is the application of a heuristic function "carried out by several agents which have a different knowledge of the task and, possibly, privacy requirements" [91]. The development of global heuristics for multi-agent scenarios must account for additional features that make heuristic evaluation an arduous task [64]:

- Solvers based on distributed computation require robust communication protocols for agents to calculate estimates for the overall task.
- For MAP approaches that preserve agents' privacy, the communication protocol must ensure that estimates are computed without disclosing sensitive private data.

The application of local or global heuristics is also determined by the characteristics of the plan synthesis scheme of the MAP solver. Particularly, in an *unthreaded* planning and coordination scheme, agents synthesize their local plans through the application of local heuristic functions. For instance, in the sequential plan synthesis scheme of MAPR [4], agents apply locally h_{FF} [42] and h_{Land} [73] when solving their allocated goals.

Local heuristic search has also been applied by some *interleaved* MAP solvers. Agents in MAFS and MAD-A* [66] generate and evaluate search states locally. An agent i shares a state S and local estimate $h^i(S)$ only if S is relevant to other planning agents. Upon reception of S, agent j performs its local evaluation of S, $h^j(S)$. Then, depending on the characteristics of the heuristic, the final estimate of S by agent j will be either $h^i(S)$, $h^j(S)$, or a combination of both. In Reference [64], authors test MAD-A* with two different optimal heuristics, *LM-Cut* [40] and *Merge&Shrink* [41], both locally applied by each agent. Despite heuristics being applied only locally, MAD-A* is proven to be cost-optimal.

Unlike unthreaded solvers, the interleaved planning and coordination strategy makes it possible to accommodate global heuristic functions. In this case, agents apply some heuristic function on their tasks $\mathcal{T}^i$ and then they exchange their local estimates to come up with an estimate for the global task $\mathcal{T}$.

GPPP [59] introduces a distributed version of a privacy-preserving *landmark* extraction algorithm for MAP. A landmark is a proposition that must be satisfied in every solution plan of a MAP task [43]. The quality of a plan in GPPP is computed as the sum of the local estimates of the agents in $\mathcal{AG}$. GPPP outperforms MAFS thanks to the accurate estimates provided by this landmark-based heuristic. MH-FMAP [91], the latest version of FMAP, introduces a multi-heuristic alternation mechanism based on Fast Downward (FD) [39]. Agents alternate two global heuristics when expanding a node in their tasks: h_{DTG}, which draws upon the information of the *Domain Transition Graphs* [38] associated with the state variables of the task, and the landmark-based heuristic h_{Land}, which only evaluates the *preferred successors* [91]. Agents jointly build the DTGs and the landmarks graph of the task and each of them stores its own version of the graphs according to its knowledge of the MAP task.

Some recent work in the literature focuses on the adaptation of well-known single-agent heuristic functions to compute global MAP estimators. The authors of Reference [83] adapt the single-agent heuristic h_{FF} [42] by means of a compact structure, the *exploration queue*, that optimizes the number of messages exchanged among agents. This multi-agent version of h_{FF}, however, is not

Table 3. Categorization of Privacy Properties in MAP

Privacy criterion	Categories
Modelling of private information	Imposed privacy [9]
	Induced privacy [90]
Information sharing	MA-STRIPS [9]
	Subset privacy [3]
Practical guarantees	No privacy [20]
	Weak privacy [4]
	Object cardinality privacy [78]
	Strong privacy [7]

as accurate as the single-agent one. The work in Reference [82] introduces a global MAP version of the admissible heuristic function *LM-Cut* that is proven to obtain estimates of the same quality as the single-agent *LM-Cut*. This multi-agent *LM-Cut* yields better estimates at the cost of a larger computational cost.

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 approaches prove the potential of the development and combination of global heuristics toward scaling up the performance of MAP solvers.

4.6 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 the task $\mathcal{T}_1$ of Section 3.2, which includes two different agents, $ta1$ and $ta2$, both representing a transport agency. Although both agents are meant to cooperate for solving this task, it is unlikely that they are willing to reveal sensitive information to a potential competitor.

Privacy in MAP has been mostly neglected and under-represented in the literature. Some paradigms like Hierarchical Task Network (HTN) planning apply an form of implicit privacy when an agent delegates subgoals to another agent, which solves them by concealing the resolution details from the requester agent. This makes HTN a very well-suited approach for practical applications like composition of web services [80]. However, formal treatment of privacy is even more scarce. One of the first attempts to come up with a formal privacy model in MAP is found in Reference [96], where authors quantify privacy in terms of the Shannon's information theory [79]. 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 issue in MAP, as it is in distributed constraint satisfaction problems [30]. A more recent work, also based on Shannon's information theory [85], quantifies privacy leakage for MA-STRIPS according to the reduction of the number of possible transition systems caused by the revealed information. In this work, the main sources of privacy leakage are identified but not experimentally evaluated.

The next subsections analyze the privacy models adopted by the MAP solvers in the literature according to three different criteria (see summary in Table 3): the *modelling* of private information, the *information sharing* schemes, and the privacy *practical privacy guarantees* offered by the MAP solver.

4.6.1 Modelling of Private Information. This feature is closely related to whether the language used to specify the MAP task enables explicit modelling of privacy or not.

Early approaches to MAP, such as MA-STRIPS [9], manage a notion of *induced privacy*. Since the *MA-STRIPS* language does not explicitly model private information, the agents' private data are inferred from the task structure. Given an agent $i \in \mathcal{AG}$ and a piece of information $p^i \in \mathcal{T}^i$, p^i is defined as private if $\forall_{j \in \mathcal{AG} | j \neq i} p^i \notin \mathcal{T}^j$; that is, if p^i is known to i and ignored by the rest of agents in $\mathcal{T}$.

FMAP [90] introduces a more general *imposed privacy* scheme, explicitly describing the private and shareable information in the task description. *MA-PDDL* [52], the language used in the CoDMAP competition [51], follows this imposed privacy scheme and allows the designer to model the private elements of the agents' tasks.

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

4.6.2 Information Sharing. Privacy-preserving algorithms vary accordingly to the number of agents that share a particular piece of information. In general, we can identify two information sharing models, namely MA-STRIPS and *subset privacy*.

MA-STRIPS . The MA-STRIPS information sharing model [9] defines as public the data that are shared among all the agents in $\mathcal{AG}$, so a piece of information is either known to all the participants, or only to a single agent. More precisely, a proposition $p \in \mathcal{P}$ is defined as *internal* or *private* to an agent $i \in \mathcal{AG}$ if p is only used and affected by the actions of $\mathcal{A}^i$. However, if the proposition p 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 p is publicly accessible to all the agents in $\mathcal{AG}$.

An action $\alpha \in \mathcal{A}^i$ that contains only public preconditions and effects is said to be public and it is known to all the participants in the task. In case α includes both public and private preconditions and effects, agents share instead α_p, the *public projection* of α, an abstraction that contains only the public elements of α.

This simple dichotomic privacy model of MA-STRIPS does not allow for specifying MAP tasks that require some information to be shared only by a subset of the planning agents in $\mathcal{AG}$.

Subset Privacy. Subset privacy is introduced in Reference [3] and generalizes the MA-STRIPS scheme by establishing pairwise privacy. This model defines a piece of information as private to a single agent, publicly accessible to all the agents in $\mathcal{AG}$ or known to a *subset of agents*. This approach is useful in applications where agents wish to conceal some information from certain agents.

For instance, agent $ta2$ in task $\mathcal{T}_1$ of Section 3.2 notifies the factory agent ft whenever the proposition (*pos* $t2$ ft) is reached. This proposition indicates that the truck $t2$ is placed at the factory ft, a location that is known to both $ta2$ and ft. Under the MA-STRIPS model, agent $ta1$ would be notified that truck $t2$ is at the factory ft, an information that $ta2$ may want to conceal. However, the subset privacy model allows $ta2$ to hide (*pos* $t2$ ft) from $ta1$ by defining it as private between $ta2$ and ft.

Hence, subset privacy is a more flexible information sharing model compared to the more conservative and limited approach of MA-STRIPS, which enables representing more complex and realistic situations concerning information sharing.

4.6.3 Privacy Practical Guarantees. Recent studies devoted to a formal treatment of practical privacy guidelines in MAP [66, 78] conclude that some privacy schemes allow agents to infer private information from other agents through the transmitted data. According to these studies, it is possible to establish a four-level taxonomy to classify the 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. Privacy has been mostly neglected in MAP but has been extensively treated within the MAS community [86]. The 2015 CoDMAP competition introduced a more expressive definition of privacy than MA-STRIPS and this was a boost for many planners to model private data in the task descriptions. Nevertheless, we can cite a large number of planners that completely disregard the issue of privacy among agents such as early approaches like GPGP [20] or more recent approaches like μ-SATPLAN [23], A# [45], or DPGM [70].

Weak Privacy. A MAP system is weakly privacy-preserving if agents do not explicitly communicate their private information to other agents at execution time [7]. This is accomplished by either *obfuscating* (encrypting) or *occluding* the private information they communicate to other agents to only reveal the public projection of their actions. In a weak privacy setting, agents may infer private data of other agents through the information exchanged during the plan synthesis.

Obfuscating the private elements of a MAP task is an appropriate mechanism when agents wish to conceal the meaning of propositions and actions. In obfuscation, the proposition names are encrypted but the number and unique identity of preconditions and effects of actions are retained, so agents are able to reconstruct the complete isomorphic image of their tasks. In MAPR [4], PMR [56], and CMAP [5], when an agent communicates a plan, it encrypts the private information to preserve its sensitive information. Agents in MAFS [66], MADLA [84], MAPlan [32], and GPPP [59] encrypt the private data of the relevant states that they exchange during the plan synthesis.

Other weak privacy-preserving solvers in the literature *occlude* the agents' private information rather than sharing obfuscated data. Agents in MH-FMAP [91] only exchange the public projection of the actions of their partial-order plans, thus occluding private information like preconditions, effects, links, or orderings.

Object Cardinality Privacy. Recently, the DPP planner [78] introduced 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 [78]. 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 $\mathcal{T}_1$ of Section 3.2. 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 the number of trucks it possesses or the number of transport routes it uses.

Strong Privacy. A MAP algorithm is said to strongly preserve privacy if none of the agents in $\mathcal{AG}$ is able to infer a private element of an agent's task from the public information it obtains during planning. To guarantee strong privacy, it is necessary to consider several factors, such as the nature of the communication channel (synchronous, asynchronous, lossy) or the computational power of the agents.

Secure-MAFS [7] is a theoretical proposal to strong privacy that builds upon the MAFS [66] model. In Secure-MAFS, two states that only differ in their private elements are not communicated to other agents to prevent them from deducing information through the non-private or public part of the states. Secure-MAFS is proved to guarantee strong privacy for a sub-class of tasks based on the well-known *logistics* domain.

In summary, weak privacy is easily achievable through obfuscation of private data, but provides little security. On the other hand, the proposal of Secure-MAFS lays the theoretical foundations to strong privacy in MAP but the complexity analysis and the practical implementation issues of this approach have not been studied yet. Additionally, object cardinality privacy accounts for a middle ground between weak and strong privacy. In general, the vast majority of MAP methods are classified under the no privacy or weak privacy levels: the former approaches to MAP do not consider privacy at all, while most of the recent proposals, which claim to be privacy preserving, resort in most cases to *obfuscation* to conceal private information.

5 DISTRIBUTED AND MULTI-AGENT PLANNING SYSTEMS TAXONOMY

As discussed in Section 1, MAP is a long-running research field that has been covered in several articles [18, 22, 61]. This section reviews the large number of domain-independent cooperative MAP solvers that have been proposed since the introduction of the MA-STRIPS model [9]. This large body of research was recently crystallized in the 2015 CoDMAP competition [51], the first attempt to directly compare MAP solvers through a benchmark encoded using the standardized *MA-PDDL* language.

The cooperative solvers analyzed in this section cover a wide range of different plan synthesis schemes. As discussed in Section 4, one can identify several aspects that determine the features of MAP solvers; namely, *agent distribution*, *computational process*, *plan synthesis scheme*, *communication mechanism*, *heuristic search*, and *privacy preservation*. This section presents an in-depth taxonomy that classifies solvers according to their main features and analyzes their similarities and differences (see Table 4).

This section also aims to critically analyze and compare the strengths and weaknesses of the planners regarding their applicability and experimental performance. Given that a comprehensive comparison of MAP solvers was issued as a result of the 2015 CoDMAP competition, Table 4 arranges solvers according to their positions in the coverage ranking (number of problems solved) of this competition. The approaches included in this taxonomy are organized according to their plan synthesis scheme, an aspect that ultimately determines the types of MAP tasks they can solve. Section 5.1 discusses the planners that follow an unthreaded planning and coordination scheme, while Section 5.2 reviews interleaved approaches to MAP.

5.1 Unthreaded Planning and Coordination MAP Solvers

The main characteristic of unthreaded planners is that planning and coordination are not intertwined but handled as two separate and independent activities. Unthreaded solvers are labelled as *UT* in the column *Coordination strategy* of Table 4. They typically apply local single-agent planning and a combinatorial optimization or satisfiability algorithm to coordinate the local plans.

Planning First, 2008 (Implemented in 2010). Planning First [67] is the first MAP solver that builds upon the MA-STRIPS model. It is an early representative of the unthreaded strategy that inspired the development of many subsequent MAP solvers, which are presented in the next paragraphs. Planning First generates a local plan for each agent in a centralized fashion by means of the FF planner [42] 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. The DisCSP is then used to find consistent coordination

Table 4. Summary of the State-of-the-Art MAP Solvers and Their Features

MAP Solver	Coordination strategy	Computational process	Plan synthesis scheme	Heuristic	Privacy	CoDMAP	
						Cent. track	Dist. track
ADP [16]	UT	C	Automated task agentization & heuristic forward search (FD)	L	N	1st/5th	-
MAP-LAPKT [62]	UT	C	Task mapping into single-agent task & heuristic forward search (LAPKT)	G	W	2nd/3rd/6th	-
MARC [81]	UT	C	Task mapping into transformer agent task & planning via FD or IBACOP → solution plan translation into original MAP task	G	W	4th	-
CMAP [5]	UT	C	Pre-planning goal allocation → task mapping into single-agent task → solution plan parallelization & heuristic forward search (LAMA)	G	W	7th/8th	-
MAPlan [32]	IL	D	Multi-agent heuristic forward search	G/L	W	9th/18th/19th	2nd/5th/6th
GPPP [60]	IL	C	Multi-agent heuristic forward search (relaxed, subgoals)	G	W	10th/11th	-
PSM [94]	UT	D	Intersection of Finite Automata & heuristic forward search (LAMA) → Finite Automata	G	W	12th/16th	1st/4th
MADLA [83]	IL	C	Multi-agent multi-heuristic state-based search	G/L	W	13th	-
PMR [56]	UT	C	Pre-planning goal allocation → Plan merging → plan repair → solution plan parallelization & heuristic forward search (LAMA)	L	W	14th	-
MAPR [4]	UT	C	Pre-planning goal allocation → iterative response planning → solution plan parallelization & heuristic forward search (LAMA)	L	W	15th	-
MH-FMAP [91]	IL	D	Multi-agent A* multi-heuristic search via forward POP	G	W	17th	3rd
DPP [78]	UT	C	Synthesis of high-level plan over DP projection (FD) & heuristic forward search (FF)	L	OC	-	-
FMAP [90]	IL	D	Multi-agent A* heuristic search via forward POP	G	W	-	-
MAFS [64]	IL	D	Multi-agent heuristic forward search	L	W	-	-
MAD-A* [64]	IL	D	Multi-agent A* heuristic forward search	L	W	-	-
MAP-POP [89]	IL	D	Multi-agent A* heuristic search via backward POP	G	W	-	-
Planning First [9]	UT	C	Post-planning coordination via DisCSP & heuristic forward search (FF)	–	N	-	-
μ-SATPLAN [23]	UT	C	Pre-planning goal allocation → iterative response planning & SAT	–	N	-	-
TFPOP [53]	UT	C	Forward-chaining partial-order planning & synthesis of agent-specific thread of actions	–	N	-	-
Distoplan [29]	UT	C	Message passing algorithm & Finite Automata	–	N	-	-
DPGM [70]	UT	C	Iterative response planning & GraphPlan + CSP plan extraction	–	N	-	-
A# [45]	UT	C	Asynchronous communication mechanism & A* heuristic forward search	G	N	-	-
Secure-MAFS [7]	IL	C	Multi-agent heuristic forward search	L	S	-	-

For unthreaded solvers, the plan synthesis schemes are listed in the form of pairs “Agent Coordination” and “Local Planning Technique.”
Computational process: C, centralized solver; D, distributed solver.
Coordination strategy: UT, unthreaded; IL, interleaved.
Privacy: N, no privacy; W, weak privacy; OC, object cardinality privacy; S, strong privacy.
Heuristic: L, local; G, global.
CoDMAP: coverage classification: Cent. track, centralized track; Dist. track, distributed track; “-” indicates that the solver did not participate in a track.

points between the local plans. If the DisCSP solver finds a solution, then the plan for the MAP task is directly built from the local plans, since the DisCSP solution guarantees compatibility among the underlying local plans.

The authors of [67] empirically evaluate Planning First over a set of tasks based on the well-known *rovers*, *satellite*, and *logistics* domains. The results show that a large number of coordination points among agents derived from the number of public actions limits the scalability and effectiveness of Planning First. Later, the MAP-POP solver outperformed Planning First in both execution time and coverage [88].

DPGM, 2010 (Implemented in 2013). DPGM [70] makes also use of CSP techniques to coordinate the agents' local plans. Unlike Planning First, 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, then DPGM backtracks to the previous agent, which puts forward an alternative plan with different coordination constraints.

μ-SATPLAN, 2010. μ-SATPLAN [23] is a MAP solver that extends the satisfiability-based planner SATPLAN [49] 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 unable to attain tasks that include group goals, because it assumes that each agent can solve its assigned goals by itself. μ-SATPLAN is experimentally validated on several multi-agent tasks of the *logistics*, *storage* and *TPP* domains. Although these tasks feature only two planning agents, the authors claim that μ-SATPLAN is capable of solving tasks with a higher number agents.

MAPR, PMR, CMAP, 2013-2015. Multi-Agent Planning by Reuse (MAPR) [4] allocates the goals $\mathcal{G}$ of the task among the agents before planning through a relaxed reachability analysis. The private information of the local plans is encrypted, thus preserving weak privacy by obfuscating the agents' local tasks. MAPR also follows an iterative response plan synthesis scheme, wherein an agent takes as input the result of the prior agent's solution plan and runs the LAMA planner [73] to obtain an extended solution plan that attains its allocated agent's goals. The plan of the last agent is a solution plan for the MAP task, which is parallelized to ensure that execution agents perform as many actions in parallel as possible. MAPR is limited to tasks that do not feature specialized agents or group 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.

Plan Merging by Reuse (PMR) [56, 57] draws upon the goal allocation and obfuscation privacy mechanisms of MAPR. Unlike MAPR, agents carry out the planning stage simultaneously instead of sequentially and each agent generates local plans for its assigned goals. In the post-planning plan merging strategy of PMR, the resulting local plans are concatenated, yielding a sequential global solution. If the result of the merging process is not a valid solution plan, then local plans are merged through a repair procedure. If a merged solution is not found, then the task is solved via a single-agent planner.

Although CMAP [5] follows the same goal allocation and obfuscation strategy of MAPR and PMR, the plan synthesis scheme of CMAP transforms the encrypted local tasks into a single-agent task ($|\mathcal{AG}| = 1$), which is then solved through the planner LAMA. CMAP was the best-performing approach of this family of MAP planners in the 2015 CoDMAP competition as shown in Table 4. CMAP ranked 7th in the centralized track and exhibited a solid performance over the 12 domains of the CoDMAP benchmark (approximately 90% coverage). PMR and MAPR ranked 14th and 15th, with roughly 60% coverage. The plan synthesis scheme of MAPR affects its performance in domains that feature group goals, such as *depots* or *woodworking*, while PMR offers a more stable performance over the benchmark.

MAP-LAPKT, 2015. MAP-LAPKT [62] conceives a MAP task as a problem that can be transformed and solved by a single-agent planner using the appropriate encoding. More precisely, MAP-LAPKT compiles the MAP task into a task that features one planning agent ($|\mathcal{AG}| = 1$) and solves with the tools provided in the repository LAPKT [71]. The authors of [62] try three different variations of best-first and depth-first search that result in algorithms with different theoretical properties and performance. The task translation performed by MAP-LAPKT offers weak privacy preservation guarantees.

As shown in Table 4, two of the three versions of MAP-LAPKT that participated in the CoDMAP ranked 2nd and 3rd in the centralized track. The coverage of MAP-LAPKT and of CMAP is roughly to 90% of the benchmark problems, an indication that shows the efficiency of the scheme that compiles a MAP task into a single-agent task.

MARC, 2015. The Multi-Agent Planner for Required Cooperation (MARC) [81] is a centralized MAP solver based on the theory of required cooperation [102]. MARC analyzes the agent distribution of the MAP task and comes up with a different arrangement of planning agents. Particularly, MARC compiles the original task into a task with a set of *transformer agents*, each one being an ensemble of various agents; that is, $|\mathcal{AG}_{MARC}| < |\mathcal{AG}|$. A transformer agent comprises the representation of various agents of the original MAP task including all their actions. The current implementation of MARC compiles all the agents in $\mathcal{AG}$ into a single transformer agent ($|\mathcal{AG}_{MARC}| = 1$). Then, a solution plan is computed via FD [39] or the portfolio planner IBACOP [10], and the resulting plan is subsequently translated into a solution for the original MAP task. MARC preserves weak privacy, since private elements of the MAP task are occluded in the transformer agent task.

Regarding experimental performance, MARC ranks at the fourth position of the centralized CoDMAP with 90% coverage, thus being one of the best-performing MAP approaches. The experimental results also reveal the efficiency of this multi-to-one agent transformation.

ADP, 2013. The Agent Decomposition-based Planner (ADP) [16] is a factored planning solver that exploits the inherently multi-agent structure (agentization) of some 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 are achievable from the current state by one of the agents. A search process, guided through the well-known h_{FF} heuristic, is then applied to find a plan that achieves these subgoals, thus resulting in a new state. This mechanism iterates successively until a solution is found.

Experimentally, ADP outperforms several state-of-the-art classical planners (e.g., LAMA) and is the top-ranked solver at the centralized track of the CoDMAP, outperforming other approaches that compile the MAP task into a single-agent planning task, such as MAP-LAPKT or CMAP.

Distoplan, 2010. Distoplan [29] is a factored planning approach that exploits independence within a planning task. Unlike other factored methods [50], 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 automaton, 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 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).

Distoplan was experimentally tested in a factored version of the *pipesworld* domain. However, the solver is unable to solve even the smallest instances of this domain in a reasonable time. The authors claim that the reason is that Distoplan scales roughly as n^3, where n is the number of components of the global task. For obvious reasons, Distoplan has not been empirically compared against other MAP solvers in the literature.

A#, 2012 (Not Implemented). In the line of factored planning, A# [45] 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 in parallel a modified version of the A* algorithm in each component, and the local search processes are guided toward finding local plans that are compatible with each other. Each local A* finds a plan as a path search in a graph and informs its neighbors 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. The authors of Reference [45] do not validate A# experimentally; however, the soundness, completeness, and optimality properties of A# are formally proven.

PSM, 2014. PSM [93, 94] is a recent distributed MAP solver that follows Distoplan's compact representation of local agents plans into Finite Automata, called Planning State Machines (PSMs). This planner defines two basic operations: obtaining a public projection of a PSM and merging two different PSMs. These operations are applied to build a public PSM consisting of merged public parts of individual PSMs. The plan synthesis 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 as it obfuscates states of the PSMs in some situations.

PSM applies an efficient handling of communication among agents, which grants this solver a remarkable experimental performance in both the centralized and distributed setting. In the centralized CoDMAP track, PSM ranks 12th (solving 70% of the tasks), and it is the top performer at the distributed track of the competition.

DPP, 2016. The DP-Projection Planner (DPP) [78], is a centralized MA-STRIPS solver that uses the Dependency-Preserving (DP) projection, a novel and accurate public projection of the MAP task information with object cardinality privacy guarantees. The single planning agent of DPP uses the FD planner to synthesize a high-level plan, which is then extended with the agents' private actions via the FF planner, thus resulting in a multi-agent solution plan.

The authors of Reference [78] provide a comprehensive experimental evaluation of DPP through the complete benchmark of the 2015 CoDMAP competition. The results show that DPP outperforms most of the top contenders of the competition (namely, GPPP, MAPR, PMR, MAPlan, and PSM). All in all, DPP can be considered the current top MA-STRIPS-based solver, as well as one of the best-performing MAP approaches to date.

TFPOP, 2011. TFPOP [53] is a hybrid approach that combines the flexibility of partial-order planning and the performance of forward-chaining search. Unlike most MA-STRIPS-based solvers, TFPOP supports temporal reasoning with durative actions. TFPOP is a centralized approach that synthesizes a solution for multiple executors. It computes *threaded partial-order plans*; that is, non-linear plans that keep a thread of sequentially ordered actions per agent, since authors assume that an execution agent performs its actions sequentially.

TFPOP is tested in a reduced set of domains, which include the well-known *satellite* and *zenotravel* domains, as well as a UAV delivery domain. The objective of this experimentation is to compare TFPOP against several partial-order planners. TFPOP is not compared to any MAP solver in this taxonomy.

5.2 Interleaved Planning and Coordination MAP Solvers

Under the interleaved scheme, labelled as *IL* in the column *Coordination strategy* of Table 4, agents jointly explore the search space intertwining their planning and coordination activities. 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. In this family of MAP solvers, agents apply a distributed exploration of the plan space. Agents locally compute plans through an embedded partial-order planning (POP) component and they build a joint search tree by following an A* search scheme guided by global heuristic functions.

MAP-POP [88, 89] performs an incomplete search based on a classical backward POP algorithm and POP heuristics. FMAP [90] introduces a sound and complete plan synthesis scheme that uses a forward-chaining POP [2] guided through the h_{DTG} heuristic. MH-FMAP [92] applies a multi-heuristic search approach that alternates h_{DTG} and h_{Land}, building a Landmark Graph (LG) to estimate the number of pending landmarks of the partial-order plans. The three planners guarantee weak privacy, since private information is occluded throughout the planning process and heuristic evaluation. The h_{Land} estimator uses some form of obfuscation during the construction of the LG.

Regarding experimental results, FMAP is proven to outperform MAP-POP and MAPR in terms of coverage over 10 MAP domains, most of which are included in the CoDMAP benchmark. Results in Reference [91] indicate that MH-FMAP obtains better coverage than both FMAP and GPPP. Interestingly, this planner exhibits a much worse performance in the CoDMAP (see Table 4), ranking 17th with only 42% coverage. This is due to the lose of accuracy of the h_{DTG} heuristic when the internal state-variable representation of the tasks in MH-FMAP is transformed to a propositional representation to be tested in the CoDMAP benchmark, thus compromising the performance of the solver [92].

MADLA, 2013. The Multiagent Distributed and Local Asynchronous Planner (MADLA) [82] is a centralized solver that runs one thread per agent on a single machine and combines two versions of the h_{FF} heuristic, a projected (local) variant (h_L) and a distributed (global) variant (h_D) in a multi-heuristic state-space search. The main novelty of MADLA is that the agent that is computing h_D, which requires contributions of the other agents for calculating the global heuristic estimator, is run asynchronously and so it can continue the search using h_L while waiting for responses from other agents that are computing parts of h_D. MADLA evaluates as many states as possible using the global heuristic h_D, which is more informative than h_L. This way, MADLA can use a computationally hard global heuristic without blocking the local planning process of the agents, thus improving the performance of the system.

Experimentally, MADLA ranks 13th in the centralized CoDMAP, reporting 66% coverage. It outperforms most of the distributed MAP solvers of the competition, but it is not able to solve the most complex tasks of the CoDMAP domains, thus not reaching the figures of the top performers such as ADP, MAP-LAPKT, or MARC.

MAFS, MAD-A, 2012-2014.* MAFS [66] is an updated version of Planning First that implements a distributed algorithm wherein agents apply a heuristic state-based search (see Section 4.5). In Reference [64], authors present MAD-A*, a cost-optimal variation of MAFS. 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* [40] and the abstraction heuristic *Merge&Shrink* [41]. MAD-A* is the first distributed and interleaved solver based on MA-STRIPS.

MAFS is compared against MAP-POP and Planning First over the *logistics*, *rovers* and *satellite* domains, notably outperforming both solvers in terms of coverage and execution time [66]. On the other hand, the authors of Reference [66] only compare MAD-A* against single-agent optimal solvers.

Secure-MAFS, 2015 (Not Implemented). Secure-MAFS [7] is an extension of MAFS toward secure MAP, and it is currently the only solver that offers *strong privacy* guarantees (see Section 4.6.3). Currently, Secure-MAFS is a theoretical work that has not been yet implemented nor experimentally evaluated.

GPPP, 2014. The Greedy Privacy-Preserving Planner (GPPP) [59, 60] builds upon MAFS and improves its performance via a global landmark-based heuristic function. GPPP applies 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) and obtaining a skeleton plan. In the *local planning* stage, agents compute private plans to achieve the preconditions of the actions in the skeleton plan. Since 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.

GPPP provides a notable experimental performance, ranking 10th in the centralized CoDMAP track. GPPP reaches 83% coverage and is only surpassed by the different versions of ADP, MARC, MAP-LAPKT, CMAP and MAPlan, which proves the accuracy of its landmark based heuristic and the overall efficiency of its plan synthesis scheme.

MAPlan, 2015. MAPlan [32] 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 under different configurations. MAPlan can be executed in a single-machine, using local communication, or in a distributed fashion, where each agent is in a different machine and communication among agents is implemented through network message passing. Regarding privacy, MAPlan applies a form of obfuscation, replacing private facts in search states by unique local identifiers, which grants weak privacy.

MAPlan exhibits a very solid performance in the centralized and distributed tracks of the 2015 CoDMAP competition, ranking 9th and 2nd, respectively. In the centralized track, MAPlan obtains 83% coverage, outperforming GPPP and reaching similar figures than the top-performing centralized solvers.

6 CONCLUSIONS

The purpose of this article is to comprehensively survey the state of the art in cooperative MAP, offering an in-detail overview of this rapidly evolving research field, which has experienced multiple key advances over the last decade. These contributions crystallized in the 2015 CoDMAP competition, where MAP solvers were compared through an exhaustive benchmark testing encoded with *MA-PDDL*, the first standard modelling language for MAP tasks.

In this article, the topic of MAP was studied from a twofold perspective: from the representational structure of a MAP task and from the problem-solving standpoint. We formally defined a MAP task following the well-known MA-STRIPS model and provided several examples that illustrate the features that distinguish MAP tasks from the more compact single-agent planning tasks. We also presented the modelling of these illustrative tasks with *MA-PDDL*.

MAP is a broad field that allows for a wide variety of problem-solving approaches. For this reason, we identified and thoroughly analyzed the main aspects that characterize a solver, from the architectural design to the practical features of a MAP tool. Among others, these aspects include the computational process of the solvers and the plan synthesis schemes that stem from the particular combination of planning and coordination applied by MAP tools, as well as other key features, such as the communication mechanisms used by the agents to interact with each other and the privacy guarantees offered by the existing solvers.

Finally, we compiled and classified the existing MAP techniques according to the aforementioned criteria. The taxonomy of MAP techniques presented in this survey prioritizes recent domain-independent techniques in the literature. Particularly, we focused on the approaches that took part in the 2015 CoDMAP competition, comparing their performance, strengths and weaknesses. The classification aims to provide the reader with a clear and comprehensive overview of the existing cooperative MAP solvers.

The body of work presented in this survey constitutes a solid foundation for the ongoing and future scientific development of the MAP field. Following, we summarize several research trends that have recently captured the attention of the community.

Theoretical Properties. The aim of the earlier cooperative MAP solvers was to contribute with a satisficing approach capable of solving a relatively small number of problems in a reasonable time but without providing any formal properties [4, 67]. The current maturity of the cooperative MAP field has witnessed the introduction of some models that focus on granting specific theoretical properties, such as completeness [90], optimality [66], or stronger privacy preservation guarantees [7, 78].

Privacy. The state of the art in MAP shows a growing effort in analyzing and formalizing privacy in MAP solvers. Nowadays, various approaches to model private information and to define information sharing can be found in the literature, which reveals that privacy is progressively becoming a key topic in MAP. However, the particular implementation of a MAP solver may jeopardize privacy, if it is possible for an agent to infer private information from the received public data. Aside from the four-level classification exposed in Section 4.6.3, other recent approaches attempt to theoretically quantify the privacy guarantees of a MAP solver [85]. In the same line, the authors of Reference [95] analyze the implications and limits of strong privacy and present a novel PSM-based planner that offers strong privacy guarantees.

On the other hand, one can also find work that proposes a smart use of privacy to increase the performance of MAP solvers, like DPP [78], which calculates an accurate public projection of the MAP task to obtain a robust high-level plan that is then completed with private actions. This scheme minimizes the communication requirements, resulting in a more efficient search. In Reference [58], authors introduce a novel weak privacy-preserving variant of MAFS, which ensures that two agents that do not share any private variable never communicate with each other, significantly reducing the number of exchanged messages. In general, the study of privacy in MAP is gaining much attention and more and more sophisticated approaches have been recently proposed.

MAP with Self-interested Agents. The mainstream in MAP with self-interested planning agents is handling situations that involve interactive decision making with possibly conflicting interests. Game theory, the study of mathematical models of conflict and cooperation between rational self-interested agents, arises naturally as a paradigm to address human conflict and cooperation within a competitive situation. Game-theoretic MAP is an active and interesting research field that reflects many real-world situations, and thus, it has a broad variety of applications, among which we can highlight congestion games [46], cost-optimal planning [65], conflict resolution in the search of a joint plan [47], or auction systems [74].

From a practical perspective, game-theoretic MAP has been successfully applied to ridesharing problems on timetabled transport services [44]. In general, strategic approaches to MAP are very appropriate to model smart city applications like traffic congestion prevention: vehicles can be accurately modelled as rational self-interested agents that want to reach their destinations as soon as possible, but they are also willing to deviate from their optimal routes to avoid traffic congestion issues that would affect all the involved agents.

Practical Applications. MAP is being used in a great variety of 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 perform different operations. ExPlanTech, for instance, is a consolidated framework for agent-based production planning, manufacturing, simulation and supply chain management [69].

MAP has also been used to control the flow of electricity in the Smart Grid [72]. 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) [19]. This system uses DECAF, a toolkit that provides standard services to integrate agent capabilities, and incorporates the GPGP framework [54] 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 [77]. 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 [68].

Aside from the aforementioned trends, there is still a broad variety of unexplored research topics in MAP. The solvers presented in this survey do not support tasks with advanced requirements. Particularly, handling temporal MAP tasks is an unresolved matter that should be addressed in the years to come. This problem will involve both the design of MAP solvers that explicitly support temporal reasoning and the extension of *MA-PDDL* to incorporate the appropriate syntax to model tasks with temporal constraints.

Cooperative MAP, as exposed in this article, puts the focus on offline tasks, without paying much attention to the problematic of plan execution. Online planning carried out by several agents poses a series of challenges derived from the integration of planning and execution and the need to respond in complex, real-time environments. Real-time cooperative MAP is about planning and simultaneous execution by several cooperative agents in a changing environment. This interesting and exciting research line is very relevant in applications that involve, for example, soccer robots.

Additionally, the body of work presented in this survey does not consider agents with individual preferences. Preference-based MAP is an unstudied field that can be interpreted as a middle ground between cooperative and self-interested MAP, since it involves a set of rational agents that work together toward a common goal while having their own preferences concerning the properties of the solution plan.

All in all, we believe that the steps taken in recent years toward the standardization of MAP tasks and tools, such as the 2015 CoDMAP competition or the introduction of *MA-PDDL*, will decisively contribute to foster a rapid expansion of this field in a wide variety of research directions.

ACKNOWLEDGMENTS

The authors thank the anonymous referees for their valuable and helpful feedback.

REFERENCES

[1] Eyal Amir and Barbara Engelhardt. 2003. Factored planning. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, Vol. 3. 929–935.

[2] 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'12)*. 2–10.

[3] 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'14)*. 973–974.

[4] Daniel Borrajo. 2013. Multi-agent planning by plan reuse. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS'13)*. 1141–1142.

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

[6] Craig Boutilier and Ronen I. Brafman. 2001. Partial-order planning with concurrent interacting actions. *J. Artific. Intell. Res.* 14 (2001), 105–136.

[7] 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'15)*. 1530–1536.

[8] 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.

[9] 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'08)*. 28–35.

[10] Isabel Cenamor, Tomás de la Rosa, and Fernando Fernández. 2014. IBACOP and IBACOP2 planner. In *Proceedings of the International Planning Competition (IPC'14)*.

[11] Bradley J. Clement and Edmund H. Durfee. 1999. Top-down search for coordinating the hierarchical plans of multiple agents. In *Proceedings of the 3rd Annual Conference on Autonomous Agents (AGENTS'99)*. ACM, New York, NY, 252–259.

[12] Daniel D. Corkill. 1979. Hierarchical planning in a distributed environment. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence (IJCAI'79)*. 168–175.

[13] Jeffrey S. Cox and Edmund H. Durfee. 2004. Efficient mechanisms for multiagent plan merging. In *Proceedings of the 3rd Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*. 1342–1343.

[14] Jeffrey S. Cox and Edmund H. Durfee. 2009. Efficient and distributable methods for solving the multiagent plan coordination problem. *Multiagent Grid Syst.* 5, 4 (2009), 373–408.

[15] Matthew Crosby, Anders Jonsson, and Michael Rovatsos. 2014. A single-agent approach to multiagent planning. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*. 237–242.

[16] 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'13)*. 46–54.

[17] Mathijs de Weerdt, André Bos, Hans Tonino, and Cees Witteveen. 2003. A resource logic for multi-agent plan merging. *Ann. Math. Artific. Intell.* 37, 1-2 (2003), 93–130.

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

[19] Keith Decker, Salim Khan, Carl Schmidt, Gang Situ, Ravi Makkena, and Dennis Michaud. 2002. BioMAS: A multi-agent system for genomic annotation. *Int. J. Coop. Info. Syst.* 11, 3 (2002), 265–292.

[20] Keith Decker and Victor R. Lesser. 1992. Generalizing the partial global planning algorithm. *Int. J. Coop. Info. Syst.* 2, 2 (1992), 319–346.

[21] Marie desJardins and Michael Wolverton. 1999. Coordinating a distributed planning system. *AI Mag.* 20, 4 (1999), 45–53.

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

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

[24] Jürgen Dix, Héctor Muñoz-Avila, Dana S. Nau, and Lingling Zhang. 2003. IMPACTing SHOP: Putting an AI planner into a multi-agent environment. *Ann. Math. Artific. Intell.* 37, 4 (2003), 381–407.

[25] Edmund H. Durfee. 1999. *Distributed Problem Solving and Planning*, Gerhard Weiss (ed.). MIT Press,118–149.

[26] Edmund H. Durfee and Victor Lesser. 1991. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Trans. Syst. Man Cybernet. Spec. Issue Distrib. Sensor Netw.* 21, 5 (1991), 1167–1183.

[27] Eithan Ephrati and Jeffrey S. Rosenschein. 1994. Divide and conquer in multi-agent planning. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94).* 375–380.

[28] Eithan Ephrati and Jeffrey S. Rosenschein. 1997. A heuristic technique for multi-agent planning. *Ann. Math. Artific. Intell.* 20, 1–4 (1997), 13–67.

[29] 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'10).* 65–72.

[30] 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'08)*, Vol. 2. IEEE, 350–358.

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

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

[33] Foundation for Intelligent Physical Agents. 2002. FIPA Interaction Protocol Specification. Retrieved from http://www.fipa.org/repository/ips.php3.

[34] Maria Fox and Derek Long. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *J. Artific. Intell. Res.* 20 (2003), 61–124.

[35] 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).

[36] Malik Ghallab, Dana Nau, and Paolo Traverso. 2004. *Automated Planning. Theory and Practice.* Morgan Kaufmann.

[37] Barbara J. Grosz, Luke Hunsberger, and Sarit Kraus. 1999. Planning and acting together. *AI Mag.* 20, 4 (1999), 23–34.

[38] Malte Helmert. 2004. A planning heuristic based on causal graph analysis. *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS'04)*, 161–170.

[39] Malte Helmert. 2006. The fast downward planning system. *J. Artific. Intell. Res.* 26, 1 (2006), 191–246.

[40] 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'09).* 162–169.

[41] 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'07).* 176–183.

[42] Jörg Hoffmann and Bernhard Nebel. 2001. The FF planning system: Fast planning generation through heuristic search. *J. Artific. Intell. Res.* 14 (2001), 253–302.

[43] Jörg Hoffmann, Julie Porteous, and Laura Sebastiá. 2004. Ordered landmarks in planning. *J. Artific. Intell. Res.* 22 (2004), 215–278.

[44] Jan Hrncír, Michael Rovatsos, and Michal Jakob. 2015. Ridesharing on timetabled transport services: A multiagent planning approach. *J. Intell. Transportat. Syst.* 19, 1 (2015), 89–105.

[45] 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'12).* 7377–7382.

[46] Anders Jonsson and Michael Rovatsos. 2011. Scaling up multiagent planning: A best-response approach. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS'11).* AAAI, 114–121.

[47] Jaume Jordán and Eva Onaindia. 2015. Game-theoretic approach for non-cooperative planning. In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI'15).* 1357–1363.

[48] Froduald Kabanza, Lu Shuyun, and Scott Goodwin. 2004. Distributed hierarchical task planning on a network of clusters. In *Proceedings of the 16th International Conference on Parallel and Distributed Computing and Systems (PDCS'04).* 139–140.

[49] Henry A. Kautz. 2006. Deconstructing planning as satisfiability. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 21, 1524.

[50] 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'07).* 1942–1947.

[51] Antonín Komenda, Michal Stolba, and Daniel L. Kovacs. 2016. The international competition of distributed and multiagent planners (CoDMAP). *AI Mag.* 37, 3 (2016), 109–115.

[52] Daniel L. Kovacs. 2012. A multi-agent extension of PDDL3.1. In *Proceedings of the 3rd Workshop on the International Planning Competition (IPC'12).* 19–27.

[53] 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'11).* AAAI, 138–145.

[54] Victor Lesser, Keith Decker, Thomas Wagner, Norman Carver, Alan Garvey, Bryan Horling, Daniel Neiman, Rodion Podorozhny, M. Nagendra Prasad, Anita Raja, Regis Vincent, Ping Xuan, and X. Q. Zhang. 2004. Evolution of the GPGP/TAEMS domain-independent coordination framework. *Auton. Agents Multi-Agent Syst.* 9, 1–2 (2004), 87–143.

[55] Derek Long, Henry Kautz, Bart Selman, Blai Bonet, Hector Geffner, Jana Koehler, Michael Brenner, Joerg Hoffmann, Frank Rittinger, Corin R. Anderson, Daniel S. Weld, David E. Smith, Maria Fox, and Derek Long. 2000. The AIPS-98 planning competition. *AI Mag.* 21, 2 (2000), 13–33.

[56] 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'14).* 38–44.

[57] 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.

[58] Shlomi Maliah, Ronen I. Brafman, and Guy Shani. 2017. Increased privacy with reduced communication in multi-agent planning. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17).* 209–217.

[59] Shlomi Maliah, Guy Shani, and Roni Stern. 2014. Privacy preserving landmark detection. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14).* 597–602.

[60] Shlomi Maliah, Guy Shani, and Roni Stern. 2016. Collaborative privacy preserving multi-agent planning. *Auton. Agents Multi-Agent Syst.* (2016), 1–38.

[61] Felipe Meneguzzi and Lavindra de Silva. 2015. Planning in BDI agents: A survey of the integration of planning algorithms and agent reasoning. *Knowl. Eng. Rev.* 30, 1 (2015), 1–44.

[62] 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.

[63] Dana S. Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. 2003. SHOP2: An HTN planning system. *J. Artific. Intell. Res.* 20 (2003), 379–404.

[64] 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'12).* 1265–1266.

[65] Raz Nissim and Ronen I. Brafman. 2013. Cost-optimal planning by self-interested agents. In *Proceedings of the 27th Conference on Artificial Intelligence (AAAI'13).*

[66] Raz Nissim and Ronen I. Brafman. 2014. Distributed heuristic forward search for multi-agent planning. *J. Artific. Intell. Res.* 51 (2014), 293–332.

[67] 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'10).* 1323–1330.

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

[69] Michal Pechoucek, Martin Rehák, Petr Charvát, Tomáš Vlcek, and Michal Kolar. 2007. Agent-based approach to mass-oriented production planning: Case study. *IEEE Trans. Syst. Man Cybernet. Part C* 37, 3 (2007), 386–395.

[70] Damien Pellier. 2010. Distributed planning through graph merging. In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence (ICAART'10).* 128–134.

[71] Miquel Ramirez, Nir Lipovetzky, and Christian Muise. 2015. Lightweight Automated Planning ToolKiT. Retrieved from http://lapkt.org/.

[72] 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'11).* 1446–1451.

[73] Silvia Richter and Matthias Westphal. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artific. Intell. Res.* 39, 1 (2010), 127–177.

[74] Valentin Robu, Han Noot, Han La Poutré, and Willem-Jan van Schijndel. 2011. A multi-agent platform for auction-based allocation of loads in transportation logistics. *Expert Syst. Appl.* 38, 4 (2011), 3483–3491.

[75] Óscar Sapena, Eva Onaindia, Antonio Garrido, and Marlene Arangú. 2008. A distributed CSP approach for collaborative planning systems. *Eng. Appl. Artific. Intell.* 21, 5 (2008), 698–709.

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

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

[78] 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'16).* 221–229.

[79] Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Tech. J.* 27, 3 (1948), 379–423.

[80] Evren Sirin, Bijan Parsia, Dan Wu, James Hendler, and Dana Nau. 2004. HTN planning for web service composition using SHOP2. *J. Web Semant.* 1, 4 (2004), 377–396.

[81] 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.

[82] 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'15).* 211–219.

[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'14).* 298–306.

[84] 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.

[85] Michal Štolba, Jan Tožička, and Antonín Komenda. 2016. Quantifying privacy leakage in multi-agent planning. *Proceedings of the 4rd ICAPS Workshop on Distributed and Multi-Agent Planning (DMAP'16).* 80–88.

[86] Jose M. Such, Ana García-Fornes, Agustín Espinosa, and Joan Bellver. 2012. Magentix2: A privacy-enhancing agent platform. *Eng. Appl. Artific. Intell.* (2012), 96–109.

[87] Milind Tambe. 1997. Towards flexible teamwork. *J. Artific. Intell. Res.* 7 (1997), 83–124.

[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'12)*, Vol. 242. IOS Press, 762–767.

[89] Alejandro Torreño, Eva Onaindia, and Óscar Sapena. 2014. A flexible coupling approach to multi-agent planning under incomplete information. *Knowl. Info. Syst.* 38, 1 (2014), 141–178.

[90] Alejandro Torreño, Eva Onaindia, and Óscar Sapena. 2014. FMAP: Distributed cooperative multi-agent planning. *Appl. Intell.* 41, 2 (2014), 606–626.

[91] 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'15).* 225–233.

[92] 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.

[93] Jan Tožička, Jan Jakubuv, and Antonín Komenda. 2015. PSM-based planners description for CoDMAP 2015 competition. In *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP'15).* 29–32.

[94] Jan Tožička, Jan Jakubuv, Antonín Komenda, and Michal Pěchouček. 2015. Privacy-concerned multiagent planning. *Knowl. Info. Syst.* 48, 3 (2016), 581–618.

[95] Jan Tožička, Michal Štolba, and Antonín Komenda. 2017. The limits of strong privacy preserving multi-agent planning. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17).* 221–229.

[96] 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'07).* 168–174.

[97] Roman van der Krogt. 2009. Quantifying privacy in multiagent planning. *Multiagent Grid Syst.* 5, 4 (2009), 451–469.

[98] David E. Wilkins. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann.

[99] David E. Wilkins and Karen L. Myers. 1998. A multiagent planning architecture. In *Proceedings of the 4th International Conference on Artificial Intelligence Planning Systems (AIPS'98).* 154–162.

[100] Michael Wolverton and Marie desJardins. 1998. Controlling communication in distributed planning using irrelevance reasoning. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98).* 868–874.

[101] Michael Wooldridge. 1997. Agent-based software engineering. *IEEE Proc. Softw. Eng.* 144, 1 (1997), 26–37.

[102] 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 August 2016; revised July 2017; accepted July 2017