CrossMark

Efficient approaches for multi-agent planning

Daniel Borrajo[1] · **Susana Fernández**[1]

Abstract Multi-agent planning (MAP) deals with planning systems that reason on long-term goals by multiple collaborative agents which want to maintain privacy on their knowledge. Recently, new MAP techniques have been devised to provide efficient solutions. Most approaches expand distributed searches using modified planners, where agents exchange public information. They present two drawbacks: they are planner-dependent; and incur a high communication cost. Instead, we present two algorithms whose search processes are monolithic (no communication while individual planning) and MAP tasks are compiled such that they are planner-independent (no programming effort needed when replacing the base planner). Our two approaches first assign each public goal to a subset of agents. In the first *distributed* approach, agents iteratively solve problems by receiving plans, goals and states from previous agents. After generating new plans by reusing previous agents' plans, they share the new plans and some obfuscated private information with the following agents. In the second *centralized* approach, agents generate an obfuscated version of their problems to protect privacy and then submit it to an agent that performs centralized planning. The resulting approaches are efficient, outperforming other state-of-the-art approaches.

1 Introduction

Multi-agent systems are being used for many applications. One of the newer research areas with a huge potential is the one related to integrating multi-agent systems with classical planning capabilities, generating the subfield of multi-agent planning (MAP). Automated planning deals with the task of finding sequences of actions that achieve a set of goals from

✉ Daniel Borrajo
  dborrajo@ia.uc3m.es

1 Departamento de Informática, Universidad Carlos III de Madrid, Av. Universidad 30, Leganés, Madrid, Spain

an initial state. A key requirement for most automated planning techniques is to be domain-independent, so the same problem solver is able to generate solutions in different domains by receiving as input a domain model expressed in a declarative language, PDDL (Planning Domain Description Language) [36]. Automated planning has been useful in many real-world applications, ranging from rovers' operations [1] to transportation logistics [34] or fire extinction [29]. In most of these applications, planning involves selecting plans for a set of agents (e.g., rovers, trucks or firemen). Recently, there has been an interest on handling these classical planning tasks by explicitly considering the characteristic of being multi-agent. Also, the new approaches deal with the task of preserving privacy of information among the agents.

There have been two main approaches to solve multi-agent planning tasks: centralized and distributed [24,26]. The centralized approach aims at generating the complete plan *for* all agents in the same common search episode. A problem with this approach is that the complexity grows exponentially with the number of agents in general. Also, the naïve centralized approach could have some difficulties maintaining the privacy on some internal agents' knowledge. This private information can refer to any of the planning components: states, goals and actions. For instance, in a transportation logistics application, a company might have divided its operation in several branches [34]. Each branch might receive its list of specific services (goals) to be addressed. The central branch might as well receive some common services to be planned for. So, there is a mixture of public and private goals. In some branches, they might use public trains to transport goods, while in others they might use private ships. Thus, there can be private and public objects as well. The same can be said about information on states. For instance, the locations of drivers of other branches.

Distributed planning consists of each agent solving its own planning task; planning is performed *by* the agents. However, given that in many domains, there are public interacting goals, it might be that one agent, ϕ_i, generates a solution that invalidates another agent's, ϕ_j, solution since it did not take into account ϕ_j's private and public goals and plans to solve them. Potential solutions are plan merging [32,50] or plan coordination [22], that can be as hard as centralized planning and are usually only useful for loosely coupled tasks [21]. If agents can achieve their goals creating few (or no) interactions with the plans of other agents, the tasks are called *loosely coupled*. As the number of interactions increases, the tasks become more *tightly coupled*. Interaction among agents' plans can be either positive—one agent achieves a set of (sub)goals that are also achieved by another agent—or negative—an agent deletes a subset of (sub)goals achieved by another agent. Brafman and Domshlak have recently shown that complexity of MAP can scale polynomially in the number of agents if some parameters related to the coupling level are fixed [13].

We are interested in a deterministic MAP setting, where we have a set of agents for which we have to find a solution to a collaborative MAP problem with private information [67]. Thus, agents are not self-interested and there is no reasoning on agents' utility. There are two main features of MAP. The first one can become an advantage for planning: if we know just a bit more information than standard planning tasks, i.e., who the agents are, then we can exploit that information to naturally decompose planning tasks. And that can lead to big improvements on planning efficiency, as other decomposition methods [11]. The second one can be seen as a disadvantage for planning: we have to keep some level of privacy among agents, such that no agent should be able to infer the private information of other agents.

In this paper, we deal with both issues. The user declares as input some information: the agents, and their privacy requirements. So, privacy is defined by the user, as opposed to other techniques where privacy is computed from the structure of the planning task [12]. Also, as opposed to other MAP techniques [12,66], the agents not only share the public

knowledge, but also their private knowledge. However, private knowledge is obfuscated to maintain privacy. Again, by sharing the obfuscated private information, we maintain privacy, but at the same time we are able to design more efficient MAP approaches than only sharing public information. We have devised several privacy preserving methods that balance the privacy level and planning efficiency.

We first propose MAPR (Multi-Agent Planning by plan Reuse). This approach occupies a middle ground between distributed and centralized planning. We have been inspired by iterative MAP techniques [42]. It starts from a PDDL description of the planning task (domain and problem) and generates an obfuscated version for each agent. Currently, we deal with PDDL2.1. Then, it first assigns a subset of public goals to each agent, while each agent might have an additional set of private goals. Afterward, MAPR calls the first agent to provide a solution (plan) that takes into account its private and public goals. MAPR iteratively calls each agent with the solutions provided by previous agents, augmented with domain and problem components needed to reproduce the solution (goals, literals from the state and actions). Each agent receives its own goals plus the goals of the previous agents. Thus, each agent solves its own problem, but taking into account the previous agents' augmented solutions. Since previous solutions might consider private information, all private information from an agent is obfuscated for the next ones. We have called it planning by reuse given that each agent reuses information from previous agents planning episodes. They pass their goals, and can reuse (or not) the actions in their plans. Since each agent receives the plan from the previous agent, that implicitly considers the solutions to all previous agents, instead of starting the search from scratch, it can also reuse the previous whole plan or only a subset of the actions. Therefore, we can use any recent work on planning by reuse [7,33]. Sharing the obfuscated private and public information of previous agents also solves the potential problem of invalidating previous agents plans, by forcing each agent to regenerate previous agents' plans or provide an alternative subplan to achieve previous agents' goals.

The second approach is a centralized version, CMAP (Centralized Multi-Agent Planning). It shares the same first step with MAPR: each agent obfuscates its initial planning task (domain and problem), and a central agent assigns a subset of public goals to some agents. This goal allocation indirectly generates a smaller set of agents (those to which CMAP has assigned goals). Then, each agent sends its obfuscated planning task to a common agent. This agent joins all problem descriptions and performs centralized planning over the obfuscated planning tasks. Experiments show that both techniques outperform state-of-the-art MAP techniques. One of the key differences of our approaches with respect to current MAP techniques lies on our focus on the division of labor. Both MAPR and CMAP explicitly reason about efficient ways of splitting the MAP task among agents, while other techniques focus on the collaboration of agents to achieve goals.

A previous version of this paper was published as an extended abstract in [4], and in a workshop [5]. Both approaches also participated in the First Competition of Distributed and Multiagent Planners (CoDMAP) [6].[1] The main contributions of this paper are:[2]

– Detailed definition of two new MAP algorithms, MAPR and CMAP, that explicitly consider (preserve) agents privacy.
– Definition of seven goal assignment strategies for both MAPR and CMAP. We report on three new ones with respect to our previous papers.
– Definition of four strategies for ordering agents. We only considered one before.

---

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

[2] Some of these issues were already described at a very high level in [6]. Here, we provide a comprehensive description of all the underlying algorithms and analysis.

– Definition (and implementation) of three methods to preserve privacy.
– Definition of two new domains, Port and Depots-robots, to test some of the properties of the algorithms. We had already introduced Port before.
– Evaluation of all the combinations of the previous algorithms and strategies against state-of-the-art MAP and centralized planners. We have greatly extended the experimentation.
– Evaluation of privacy preservation on different domains.

The paper is structured as follows. Section 2 presents the MAP task we are dealing with. Next, Sect. 3 defines the privacy we use in this paper and four different algorithms for maintaining privacy. Section 4 describes MAPR, and Sect. 5 presents CMAP. Section 6 describes the experimental setup, shows results and discusses them. Section 7 presents relevant state-of-the-art approaches. Finally, Sect. 8 draws some conclusions and presents some future work.

2 Single and multi-agent planning tasks

We deal here with multi-agent classical planning tasks. We describe the standard single-agent planning task. We start by providing a standard definition of single-agent planning task in the propositional setting. An alternative common setting nowadays is SAS+ [20]. Our approaches follow the propositional setting, so we will use it to describe our algorithms. We mention the SAS+ representation given that the base planner used for the experiments uses it as we describe later.

Definition 1 *(Single-agent classical planning task)* A single-agent classical planning task is a tuple $\Pi = \{F, A, I, G\}$, where F is a set of propositions, A is a set of instantiated actions, $I \subseteq F$ is an initial state, and $G \subseteq F$ is a set of goals.

Each action $a \in A$ is described by a set of preconditions (pre(a)) that represent literals that must be true in a state to execute the action and a set of effects (eff(a)), literals that are expected to be added (add(a) effects) or removed (del(a) effects) from the state after execution of the action. The definition of each action might also include a cost $c(a)$ (the default cost is one). The application of an action a in a state s is defined by a function γ, such that $\gamma(s, a) = (s \backslash \text{del}(a)) \cup \text{add}(a)$ if pre(a)$\subseteq s$ and s otherwise (it cannot be applied).[3] The planning task should generate as output a sequence of actions $\pi = (a_1, \ldots, a_n)$ such that if applied in order from the initial state I would result in a state s_n, where goals are true, $G \subseteq s_n$. Plan cost is commonly defined as: $C(\pi) = \sum_{a_i \in \pi} c(a_i)$. Even if MAPR and CMAP can deal with the same cost functions that current planners can, in this paper we will only report on quality measured as the plan length. Thus, $c(a_i) = 1 \; \forall a_i \in A$.

In order to represent planning tasks compactly, the automated planning community uses the standard language PDDL [36]. A planning task Π is automatically generated from the PDDL description of a domain D and a problem P. The domain in PDDL is a tuple $D = \{Ty, Co, Pr, Fn, Op\}$, where: Ty is a hierarchy of types (to characterize the problem objects); Co is a set of constants that are used by all problems in the domain; Pr and Fn are sets of definitions of predicates and functions, respectively, whose instantiations generate the facts in F; and Op is a set of operator schemas or generalized actions, defined using variables—parameters, par(a). The instantiations of those operators with problem objects generate the actions in A. A planning problem in PDDL is a tuple $P = \{D, Ob, I, G, Me\}$,

---

3 Also defined as ∅ when the action cannot be applied.

where: D is the domain; Ob is a set of objects (instances of types in the domain); I is the initial state; G is the set of goals; and Me is an optional metric to define optimization criteria (most commonly minimizing plan cost). We will present some results related to the quality of the solutions that will use a cost metric.

In MAP, a set of m agents, $\Phi = \{\phi_1, \ldots, \phi_m\}$ have to solve the planning task Π.

Definition 2 *(MAP task)* A MAP task is a set of planning subtasks, one for each agent, $M = \{\Pi_1, \ldots, \Pi_m\}$. Each planning subtask can be defined as a single-agent planning task, $\Pi_i = \{A_i, F_i, I_i, G_i\}$. An alternative equivalent lifted representation of each single-agent planning task in PDDL would be a pair (domain,problem): $\Pi_i = \{D_i, P_i\}$.

In the definition, we do not require the sets A_i to be disjoint. So, there can be domains where agents share a subset of actions. For instance, in the Driverlog, if we set the agents to be the drivers (a common way to *agentify*—model—the domain), there are actions (load and unload) that do not have an agent in their parameters. In those cases, these actions will be in the set of all agents. As we will see, this is a major difference in terms of modeling with other MAP approaches that assume that all actions have to be performed by at least one agent (an agent has to be in the parameter list of each action), and thus they require that the sets A_i are disjoint [12,54].

The components of each Π_i have a public part that can be shared with other agents, and a private part. We assume that both the complete initial state, $I = \cup_{i=1}^{m} I_i$, and set of goals, $G = \cup_{i=1}^{m} G_i$ are consistent; that is, they are conflict-free (there are no mutex). In other MAP approaches, they allow conflicts among goals [70].

3 Preserving privacy

One of the key requirements in MAP is privacy preservation. How to represent privacy and what levels of privacy preservation are available is still an open issue. We will first address some definitions of privacy preservation and later we will define the methods we propose and use in our MAP algorithms. Nissim and Brafman [54] define *weak privacy preserving* (*wpp*) planning algorithms as those that do not exchange private information among the agents and *strong privacy preserving* (*spp*) algorithms as those where agents cannot infer more isomorphic models of other agents information than the ones that can directly be inferred from the public information. Their approaches lie in between these two extremes. Recent work has defined formally privacy leakage in terms of the inferred transition systems [64]. Our approach is based on the idea of actually sharing information and still being able to preserve privacy in a level equivalent to Nissim and Brafman, as we will discuss later. We defined in previous works a simple way to preserve privacy [5]. Here, we define several methods for preserving privacy with different properties in relation to privacy preservation. Next, we discuss them in more detail.

3.1 Definition of privacy

The best alternative for defining MAP tasks would be to have a standard language as PDDL for single-agent planning tasks. However, the community is still small and there has been yet no agreement on such standard, even if there have already been some attempts at defining it, as MAPL [15], MA- PDDL [44] (as the one used in the recent MAP comparison CoDMAP, or the one used by Torreño et al. [68]). Most of these approaches require that the user manually

augments the PDDL (or equivalent) definitions of domains and problems to incorporate the extra multi-agent components into the corresponding language.

The most used approach, MA- STRIPS, defines a multi-agent model as a rewriting of the single-agent task [12]. In this case, the multi-agent task is redefined as $M = \{F, \cup_i A_i, I, G\}$, where each A_i is the set of actions of agent ϕ_i. Systems using MA- STRIPS only take as input the set of agents. Then, instantiated actions are assigned to the corresponding agent, and the notion of private atoms is inferred as the ones that are only handled by one agent's actions. So, in their privacy model, the focus is on defining agents' actions and then inferring the privacy of atoms. The fact that an atom (piece of information) is private is a side effect of the fact that an action belongs to an agent.

We follow a different approach. We prefer to attach the property of privacy to the information available to agents (states, goals and objects). While MA- STRIPS defines privacy at the instantiated level (propositions), we define it at the lifted representation level (PDDL). Intuitively, and also from our experience on projects, privacy relates to knowledge (state or goals) that each agent does not want to share openly with the rest of agents when planning. In particular, we can attach the property of private to predicates and types, and they will be inherited by their instantiations (atoms and objects). A side effect of our approach is that we can deal with private goals where other approaches cannot in their current state without further reformulation, and we can deal with actions with no associated agent.

To showcase a difference between the two models, suppose, for instance, a logistics domain where a company is structured in some branches, one per city. Each branch owns some trucks to move packages within the same city, and there are common airplanes that can carry packages from one city to another. Assume that, in a given problem, a truck, `truck1`, is the only one that can move a package p1 from location `locA` to location `locB` (it is the only truck in the city of both locations). MA- STRIPS would assign action `move(truck1,p1,locA,locB)` to `truck1`. Then, it would infer that the location of p1 at `locA` is private to `truck1`. However, in the real world, sometimes the location of the package might be considered as public (when we want the user to be able to track the position of the package) and sometimes as private (internal) by the whole company. In our case, the user can decide at planning time to define the predicate `at` as either private or public. So, in the case of MA- STRIPS, the definition of privacy is operational for decomposing the problem into subproblems and deciding what to share or not with the rest of agents. Instead, our definition of privacy relates to the common concept used by most people/organizations, and can be parameterized by the user by setting the predicates as private or public, as explained next.

In the experiments, we have used as agents the standard setting in most MAP works (e.g., rovers, satellites, or aircrafts), and that selection of agents leads to use as private predicates and types exactly the same ones that MA- STRIPS would generate. So, in most cases, both definitions generate the same privatization model. Torreño et al. [68] use a richer model of privacy that allows the user to specify which predicate is public for each agent. Bonisoli et al. [3] also describe a richer model of privacy where agents might not need to know the presence of other agents. In these three privacy models, the richer the model is, the more inputs it requires from the user.

Since PDDL does not allow us to define privacy, nor agents, and we wanted to minimally change the input descriptions of domains and problems from the single-agent case, we have developed a compiler that translates any classical single-agent planning task into a multi-agent planning task. The compiler takes as input a domain D and problem P in standard PDDL. In order to define the agents and the privacy level, the compiler also takes as input three lists: the agents' domain types AT, the private predicates and functions PP, and the

private types PT. We will call *agentification* of a domain to the particular assignment of values to these three variables. The compiler generates as output a domain and problem file for each agent that corresponds to the lifted representation of the MAP task of Definition 2. For example, in the logistics domain mentioned before, AT={truck}, PP={at} and PT={}. If trucks would have any measurement instrument, then PT could be {instrument}, as it would be private for each truck. AT, PP and PT are mainly used for creating the domain and problem definitions from the point of view of each agent. In particular, AT is used to guide the processes that define the particular domain and problem for each agent, and the ones that try to preserve privacy as explained later.

The domain file of a given agent contains only those actions that can be carried out by that agent or those actions that are not carried out by any agent. That is, if agent's type is t, those actions that contain a parameter of type t' such that either $t' = t$, or t' is a super-type of t in the PDDL types' hierarchy, or those actions that do not contain any parameter of a type in AT. The problem file of a given agent contains only the parts of the state and goals that are either public or private for that agent. Thus, it removes all references of objects of types in $AT \cup PT$ of one agent from the domain and problem files of the other agents.

A constraint that users should take into account is that the definition of those inputs has to be consistent. Thus, if two agents can modify the value of the same grounded literal, the corresponding predicate cannot be private; i.e., it cannot be in PP. In our experience, defining these inputs so that this property holds is really easy for the domains we have used in the experiments. Appendix A.1 shows the settings we have used in the experiments for each domain.

In case, the input domain and problem files are specified in the unfactored MA-PDDL language (as for the CoDMAP competition), then the compiler takes the input MA-PDDL domain file and problem file and defines the mapping from MA-PDDL to our model as:

– MA-PDDL private predicates: they become PP
– MA-PDDL action definition includes the agent of that action: the union of the types of those agents becomes AT
– MA-PDDL private objects: their types become PT

The additional information required to convert a single-agent problem into a multi-agent problem is the same in both cases. MA-PDDL includes it in the domain and problem files, while we provide it as a separate input to the algorithm.

Nevertheless, this compilation is only needed until we have a standard language for specifying MAP tasks. In the real world, each agent would supply its own domain and problem definitions and we would not need this compilation. Therefore, the real MAP problem solving starts in Sect. 4.

Since we will need it later, we define now public goals.

Definition 3 *(Public goal)* A goal $g \in G$ is public in the lifted (PDDL) representation if its predicate is not a private predicate, and it does not have as argument an object of a private type. The set of public goals, PG, can be defined as: $PG = \{g \mid g \in G, g = p(o_1, o_2, \ldots, o_n), p \notin PP, \nexists o_i \text{ type}(o_i) \in PT\}$, where `type()` returns the type of a given object.

We could have dropped the condition of not including a private type, but in that case the only agents that would be able to achieve it would be the "owners" of the objects of that private type. So, we directly define those goals as private.

Now, we will define three methods for privacy preservation that we have implemented: obfuscation by random substitution, generation and sharing of macro-operators, and gener-

ation of zero-arity predicates. We believe these approaches are general enough to be used by other MAP algorithms. For a better understanding of the three methods, we advance some details of our MAP algorithms described in the next section. In our MAP approaches, the search starts when each agent takes as input the compiler's output (its domain and problem files) and generates as output a first obfuscation of both files. In CMAP, a centralized planner solves the planning task generated by merging all these agents' obfuscated files. In MAPR, the first agent solves its obfuscated problem and sends to the next agent the solution plan together with domain and problem components needed to reproduce the solution. These components include the description of the actions in the solution plan in terms of preconditions and effects and the goals (where private information is obfuscated). Then, the second agent integrates the received information into its planning task and generates a plan that achieves its own goals and the previous agents' goals. The same process is repeated iteratively with the following agents.

3.2 Obfuscation by random substitution

Since information on other agents is shared in MAPR and CMAP, each agent needs to obfuscate its private information when sharing it with other agents (MAPR) or with a central agent (CMAP). Thus, each agent takes as input the compiler's output (its domain and problem files) and generates as output a first obfuscation of both files. There can be potentially many algorithms for encrypting/obfuscating the information. We define first a simple alternative. The obfuscation function, denoted as @, can be described as a three-steps process.

First, a random substitution is generated for the names of all private information. So, each agent reads its input domain and problem files and generates a random substitution σ for everything considered for obfuscation. The elements of the PDDL files that are considered for obfuscation are inferred from the inputs to the compiler, AT, PP and PT. So, they are: types in $AT \cup PT$; constants of a type in $AT \cup PT$; predicates or functions in PP or whose arguments are in $AT \cup PT$; literals from the state, goals and preconditions and effects of actions in PP; actions that include any of the previous; and objects of types in $AT \cup PT$.[4] Second, each agent ϕ_i applies its substitution to its PDDL domain and problem, generating two new files, $D_{\phi_i}^@$ and $P_{\phi_i}^@$. And, third, each agent removes from $D_{\phi_i}^@$ and $P_{\phi_i}^@$ any parameter referring to its type from actions, predicates and functions.

Let us see an example of the combined effect of the compilation and the obfuscation in a simple multi-agent robot domain (Depots-robots) inspired by the Kiva robots used by Amazon.[5] These robots move in a grid inside a depot. They have to move inventory pods from their storage places to human workers that fill orders by picking up items from the pods. In our simplified version, a set of robots and a set of pods are placed in a grid. Robots can move to adjacent cells (vertical or horizontal movements only). They can move empty or with one pod (by placing themselves under the pod and taking it). Figure 1 (left) shows an example of an initial state in this domain with two robots (R1 and R2) and three pods (P1, P2 and P3). The goals (right) would be that pods are used by humans, so they have to be at the same locations where human workers (H1–H4) are. Note that P2 cannot be moved in the initial state before moving P1 or P3. A plan to solve the problem, would move R2 to take pod P3 to H4, while robot R1 would move to take pods P1 and P2 to H1 and H3, respectively.

In Fig. 2 we show some parts of the input domain definition and their corresponding obfuscated versions for agent R1 (agents are read from the problem file). The actions would

---

[4] This step has to assure that the propositions used in the substitutions of different agents are different.
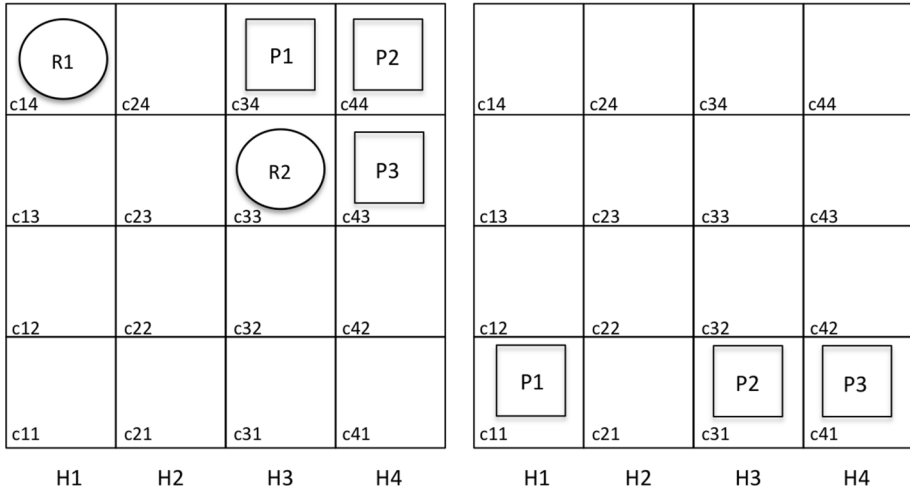
[5] www.kivasystems.com, www.amazon.com

Fig. 1 Example of an initial state (left) in a robot domain with two robots (R_i, circles), three pods (P_i, squares) and four human workers (H_i). The goals (right) would be that these three pods have been moved to service humans

```
(define (domain Depots-robots)              (define (domain Depots-robots)
  (:requirements :strips :typing)             (:requirements :strips :typing)
  (:types cell robot pod human)               (:types cell pod human)
  (:predicates                                (:predicates
    (right ?c1 - cell ?c2 - cell)               (right ?c1 - cell ?c2 - cell)
    (left ?c1 - cell ?c2 - cell)                (left ?c1 - cell ?c2 - cell)
    (up ?c1 - cell ?c2 - cell)                  (up ?c1 - cell ?c2 - cell)
    (down ?c1 - cell ?c2 - cell)                (down ?c1 - cell ?c2 - cell)
    (empty ?c - cell)                           (empty ?c - cell)
    (at-pod ?p - pod ?c - cell)                 (at-pod ?p - pod ?c - cell)
    (at-robot ?r - robot ?c - cell)             (anon1 ?c - cell)
    (free ?r - robot)                           (anon2 ?p - pod)
    (carries ?r - robot ?p - pod)               (anon0)
    (serves ?h - human ?c - cell)               (serves ?h - human ?c - cell)
    (used ?h - human ?p - pod))                 (used ?h - human ?p - pod))
  (:action move-right                         (:action anon3
   :parameters (?r - robot ?c1 - cell          :parameters (?c1 - cell ?c2 - cell)
               ?c2 - cell)                      :precondition (and (anon1 ?c1)
   :precondition (and (at-robot ?r ?c1)                            (right ?c1 ?c2)
                     (right ?c1 ?c2)                               (empty ?c2))
                     (empty ?c2))               :effect (and (anon1 ?c2)
   :effect (and (at-robot ?r ?c2)                            (empty ?c1)
               (empty ?c1)                                   (not (empty ?c2))
               (not (empty ?c2))                             (not (anon1 ?c1))))
               (not (at-robot ?r ?c1))))
  ...                                         ...
```

Fig. 2 On the left, we show some parts of the input domain file of the Depots-robots domain. On the right we show the obfuscated version for agent `R1`

be: move robots (four when robots are free and four to take the pods); dropping the pods (one action); and a human using a pod (one action). The agents in this domain are $AT =\{$robot$\}$. The private predicates are $PP=\{$at-robot, carries, free$\}$. All goals would be public. In our simplified domain, there are no private types. In a more complex scenario, we could model other issues, such as the robots' batteries, that would be private types, while the

```
(define (problem p1)
 (:domain Depots-robots)
 (:objects R1 R2 - robot            (define (problem p1)
          P1 P2 P3 - pod             (:domain Depots-robots)
          H1 H2 H3 H4 - human        (:objects P1 P2 P3 - pod
          C11 C21 C31 ... C44 - cell)         H1 H2 H3 H4 - human
 (:init (at-robot R1 C14) (at-robot R2 C33)   C11 C21 C31 ... C44 - cell)
        (free R1) (free R2)         (:init (anon1 C14) (anon0)
        (at-pod P1 C34) (at-pod P2 C44)       (at-pod P1 C34) (at-pod P2 C44)
        (at-pod P3 C43) (serves H1 C11) ...   (at-pod P3 C43) (serves H1 C11) ...
        (right C11 C21) ...                   (right C11 C21) ...
        (empty C11) ...)                      (empty C11) ...)
 (:goal (and (used H1 P1) (used H3 P2)  (:goal (and (used H1 P1) (used H3 P2)
             (used H4 P3))))                  (used H4 P3))))
```

Fig. 3 On the left we show some parts of an input problem file of the Depots-robots domain. On the right we show the obfuscated version for agent `R1`

battery level would be a private predicate/function. Figure 3 shows some parts of the problem file and their corresponding obfuscated versions.

The resulting substitution applicable to R1 is:[6]

$\sigma =\{$`(free/anon0), (at-robot/anon1), (carries/anon2)`$, …\}$

Similarly, a possible substitution applicable to R2 would be:

$\sigma =\{$`(free/nona0), (at-robot/nona1), (carries/nona2)`$, …\}$

Any stronger obfuscation function @ could be used instead, without loss of generality of the rest of the approach. The main constraint is that the technique has to maintain the standard properties of planning tasks. Examples of such properties are:

1. if the obfuscation technique converts an action $a \in A$ of an agent ϕ into its obfuscated version, $a^@$, then for all states s such that a is applicable in s, $a^@$ is also applicable in the corresponding $s^@$;
2. if a of an agent ϕ is applied in a state s, resulting in a state s_1, then if $a^@$ is applied in $s^@$, it must result in $s_1^@$

It is easy to see that the obfuscation technique proposed in this section fulfills these properties. Each action a of a given agent ϕ is mapped into one action $a^@$ where we have applied a substitution @ to its preconditions and effects, followed by removing the agent variable from all the lifted literals in pre$(a) \cup$add$(a) \cup$del(a). We also apply the same operation to I, resulting in $I^@$. Given that we perform the same mapping operation to each action a and the I, then each literal will either not change, or change in the same way in I and a. Thus, for all actions a, such that $pre(a) \subseteq I$, then $pre(a^@) \subseteq I^@$ (if they were applicable in I, the mapped action will also be applicable in $I^@$). Now, for all actions a applicable in I, the state after applying a in I is $s_1 = \gamma(I, a)$. We also applied the same mapping to the effects of a, so the result of applying $a^@$ in $I^@$, $s_2 = \gamma(I^@, a^@)$, should be $s_2 = s_1^@$. Therefore, we have shown that the properties hold in the case of the initial state. If we apply them recursively, they hold in all reachable states from I.

In relation to space complexity, the size of the input PDDL domain file, $|D|$ is similar to (slightly bigger than) the size of each of the agents' PDDL domain file, $|D| \sim |D_{\phi_i}^@|$, $i = 1..m$. We now have m such files. On the other hand, the size of each agent's PDDL problem file is usually much smaller from the original PDDL problem file $|P| \sim \frac{|P_{\phi_i}^@|}{m}$,

---

[6] All the new anonymized names, anon$X$, can be replaced by other arbitrary names.

$i = 1..m$. In the obfuscated problem file of each agent, there is only private information of that agent, plus public information. Therefore, the overhead is proportional (around m times) to the size of the public information. If we see it from each agent's perspective (the planning tasks we will actually solve in MAPR), the size of each MAP task Π_i is around m times smaller in the private part than the original one, Π. This is also relevant for the branching factor that could be reduced in a factor of m (since each action that contains an agent as parameter generates m times less instantiations), resulting in potentially exponential benefits in terms of search. In summary, one can see the compilation and obfuscation processes as a problem decomposition technique based on domain-dependent features (agent types, private predicates and private types) where space complexity is linearly augmented.

In MAPR, agents also exchange plans. Next, we will study further obfuscation techniques that can be applied when plans have been generated and improve privacy preservation. Before, we explain another type of obfuscation technique with stronger privacy-preserving properties that can be applied when plans have been computed, based on generating macro-operators.

3.3 Generation and sharing of macro-operators

In MAPR, agents will generate individual plans and share them with the next agents as discussed later on. Thus, we have devised a method to obfuscate further the information shared among agents by generating macro-operators from the agents' plans. Given a plan π, a macro-operator is defined as a new action m_π that has as preconditions those conditions of actions in the plan that are not achieved by previous actions in the plan, and whose effects are the ones that are added or deleted by the execution in sequence of the actions in π. They were defined in the STRIPS planner and have been successfully used as a learning technique [30,31]. The idea of sharing macro-operators rather than individual actions is that details on the underlying plan are missing and thus cannot be inferred by the next agents. We have defined two alternative methods that work in combination only with MAPR.

– Generate one macro-operator for the complete plan of each agent, which is communicated to the rest. Thus, if agent ϕ_i generates the plan $\pi = (a_1, \ldots, a_n)$, then we generate its corresponding macro-operator m_π using the standard procedure, though it is left fully grounded; that is, we do not perform the final generalization step [31]. Sharing the macro-operator instead of the primitive actions restricts the next agents' capability to interleave primitive actions from the previous agents in the new plan. So, it introduces a balance between privacy preservation and completeness/quality of solutions.
– Generate several macro-operators. It traverses the actions in the plan from the first one, maintaining a list of actions to be included in the next macro-operator. At each step, if action a_i contains private information, it adds it to the list of actions. If not (all information is public), then it generates a macro-operator with all the collected actions, empties the list of actions and continues. Once it finishes, it sends the next agents the list of generated macro-operators. Nissim and Brafman [54] mentioned that they tried to use a variation of this idea, but it was not useful for them due to the utility problem [53]. Very recently, a variation of that approach has been published [51]. The main difference with our proposal is that they create and use the macros online (during the search of a solution), and in MAPR agents build them after their search of a solution has finished and the macro-operators are used by the following agents.

Suppose, for instance, in a transportation problem where some trucks have to move packages from one place to another using a road network. They have three actions: load, unload, and move. The private component of the state would be the location of each truck, so loading

and unloading have a mixture of public and private information, and move is completely private. Assume truck `tr1` generates a plan such as:

```
load(t1,p1,n1), move(t1,n1,n2), move(t1,n2,n3), unload(t1,p1,n3),
move(t1,n3,n4), load(t1,p2,n4), move(t1,n4,n3), unload(t1,p2,n3)
```

Then, the first method would build and share a single macro-operator m with the following contents (that would be further obfuscated as described in Sect. 4):

```
pre(m)={at(tr1,n1),at(p1,n1),at(p2,n4),conn(n1,n2),…,conn(n4,n3)}
```[7]
```
eff(m)={at(p1,n3),at(p2,n3),not(at(tr1,n1)),at(tr1,n3)}
```

`n2` disappears from the dynamic predicates. Thus, in case, there are different ways to go from $n1$ to $n3$, the fact of having gone through `n2` would be hidden for the next agent. The second method would generate the following macro-operators:

$m_1(t1, p1, n1, n3)$=move-move-unload, $m_2(t1, n3, n4, p2)$=move-load, and

$m_3(t1, n4, n3, p2)$=move-unload. It would share the plan π=[load,m_1,m_2,m_3] plus some extra information (including the macro-operators models) as detailed in Sect. 4. Again, the fact of having gone through `n2` would be hidden for the next agent.

This second method is similar in terms of privacy preserving to that of distributed MAP approaches (FMAP [66], or MAFS [54]). In their case, every time an agent changes some public component of the state, it broadcasts that information to the rest (MAFS broadcasts states and FMAP also include information on the partial-order plan, such as causal links, preconditions and effects related to the changes). Thus, what each agent receives from the others can be seen as a sequence of states. Suppose that agent ϕ_i receives from another agent ϕ_j the sequence of states $\langle s_0, s_1, \ldots, s_n \rangle$ during a MAP search (each pair of consecutive states corresponds to the execution of a sequence of private actions of ϕ_j). ϕ_i can compute the differences between consecutive states that it has received from ϕ_j (similarly with the other agents). So, it can translate the input sequence of states into $\langle s_0, \Delta_1, \ldots, \Delta_n \rangle$, where $\Delta_i = (s_i \backslash s_{i-1}) \cup$ not $(s_{i-1} \backslash s_i)$. That is, the set of literals that are true in s_i that were not true in s_{i-1} (adds) and the set of literals that were true in s_{i-1} and are not true in s_i (dels). Each of these deltas can be seen as the effects of a macro-operator that explain the changes generated by ϕ_j's private actions applied between two consecutive public actions. The only difference with a macro-operator is that preconditions cannot be computed directly from each delta. However, since we have a set of deltas, some inductive procedure could be used to derive such preconditions [78,79]. Furthermore, in the case of FMAP, those preconditions are shared.

Therefore, the information exchanged among agents when learning and sharing a set of macro-operators can be seen as similar to the information that distributed approaches exchange, given that all agents broadcast to the rest all changes in the public state. In our case, each agent only receives as information the obfuscated plans, and reduced domain and problem descriptions of the previous agents (a subset of all agents). Given that the information of all the previous agents has been merged, the following agents are not able to decide from which previous agent they are receiving which part of the plan, and reduced domain and problem descriptions. In other MAP approaches, they do know which agent has made which changes to the public part of the state. The first method of learning macro-operators leaks even less information, since it removes all intermediate changes made in the public part of the state that are no longer true at the end of the plan (they will not appear as effects in the corresponding macro-operator).

---

[7] The predicate `conn` defines the road network through the connection between locations.

3.4 Generation of zero-arity predicates

The next privacy-preserving method is also applied by MAPR when agents share their plans with the next agent. It consists of the following steps:

– Since all actions in the plan are instantiated, we generate a second level of obfuscation, by removing public objects from the private literals in the preconditions and effects and generating a new symbol for the literal's name. Private objects were already removed from all literals in the first obfuscation. So, after this step, the only objects that are left in the action are public objects that appear in public literals. Also, we remove parameters from the action. Suppose, for instance, in a Logistics domain, there is an action a1=`drive-truck(?tr,?l1,?l2)`:

```
drive-truck(?tr,?l1,?l2)
pre={at(?tr,?l1), connected(?l1,?l2)}
eff={not(at(?tr,?l1)), at(?tr,l2)}
```

Since `at` is private, the first obfuscation process would have generated the following model for an agent $tr1$ (the one used by MAPR to search for a solution):

```
anon1(?l1,?l2)
pre={anon2(?l1), connected(?l1,?l2)}
eff={not(anon2(?l1)), anon2(?l2)}
```

Suppose now that agent $tr1$ has used an instantiation of that action in a plan, such as `anon1(A,B)`. A second obfuscation step is performed converting it into:

```
anon3()
pre={anon4, connected(A,B)}
eff={not(anon4), anon5}
```

Obviously, all appearances of `anon2(A)` in the actions in the plan would have to be obfuscated with the same substitution (`anon4`), and the same applies to `anon2(B)` for `anon5`. Thus, the semantics of drive actions should be inferred from the predicate `connected` and we deal with it in the next step.

– All static predicates (both public and private) in the preconditions of the actions are removed (as the `connected` predicate in the previous example). Since the action was applied, its preconditions were true in the initial state, in particular the static predicates. Also, no action can change their truth value, and they were already true, so they can be safely removed from the actions preconditions. As an example, the previous action would be shared with the remaining agents as follows. We can observe that this step has removed all semantics of the original action:

```
anon3()
pre={anon4}
eff={not(anon4), anon5}
```

Let us see now an example of where these previous steps do not remove completely the semantics of the original domain. Suppose we are in the Depots-robots domain and a robot has used an instance of the `move-right` action (in Fig. 2) such as `anon3(C31,C41)`. The previous steps would translate this action into:

```
anon27()
pre={anon28, empty(C41)}
```

```
eff={not(anon28), anon29, empty(C31), non(empty(C41))}
```

In this case, given that `empty` is a public literal, MAPR cannot obfuscate it (other agents might need to know whether cell C41 is empty or not). Thus, this second obfuscation step removes all semantics for actions with no public literals, and it keeps semantics proportional to the number of public predicates in the preconditions or effects of the actions. Again, the same applies to other MAP approaches (MAFS or FMAP). The other agents would see that `empty(C41)` is true and in the next state it is false, while `empty(C31)` becomes true.

- As explained before, MAPR can learn and share macro-operators, so that privacy-preserving is further augmented by not sharing the individual private actions (either between two public actions, or all). For instance, in the case of the previous example in the Logistics domain, the truck would not communicate all individual movements (private literals referring to the truck being at intermediate locations).

4 Multi-agent planning in MAPR

The main steps of the MAPR algorithm are to first assign public goals to agents and then iteratively solve each agent problem. Once an agent solves a problem, it communicates an augmented solution to the next agent where private components are obfuscated. In turn, the next agent should solve its own problem augmented with the obfuscated private part of the solution of the previous agents and the public part of those solutions.

4.1 MAPR **algorithm**

Figure 4 shows a high-level description of the algorithm. It takes as input a set of agents Φ, a MAP task (set of agents planning tasks), the lists of private predicates and types, a goal assignment strategy, an agents ordering scheme, the planner to be used by the first agent, and a second planner (it might be the same one) to be used by the following agents. The reason to use two planners is that the second planner might be a replanning system. Since all inputs and outputs are in PDDL, we can use any state-of-the-art planner. The MAPR algorithm requires the execution of a virtual central agent that orchestrates all the relevant calls. It starts by computing the set of public goals (from M, PP and PT following Definition 3) and a first ordering of agents. MAPR is then composed of eight main steps: goal assignment; second ordering of agents; first planning episode; building of augmented solution; communication of augmented solution to the next agent; merging of the information of prior agents with the current agent's planning problem; subsequent planning episodes; and termination. We will now explain in more detail each step.

4.2 Goal assignment

In the input MAP task M, each agent's problem definition includes all public goals. But it would be inefficient to devote all agents to achieve all goals, given that it is a collaborative task. Thus, given the total set of public goals PG and a set of agents Φ, MAPR first assigns a subset of goals to each agent. For each public goal $g \in PG$, each agent $\phi \in \Phi$ computes the cost of the relaxed plan, $c(g, \phi)$, from its initial state, I_i, following the well-known relaxed plan heuristic of FF [40]. The relaxed plan heuristic computes the cost of a plan that could reach the goals from the initial state without considering the deletes of actions. If the relaxed

Algorithm MAPR ($\Phi, M, PP, PT, GA, OS, FP, SP$): plan

Φ: a set of agents
M: MAP task
PP: list of private predicates
PT: list of private types
GA: goal assignment strategy
OS: agents ordering scheme
FP: first planner
SP: second planner

$PG \leftarrow \text{ComputePublicGoals}(M, PP, PT)$
Order agents according to OS
Assign subset of PG to each agent $\phi_i \in \Phi$ using GA
Order agents according to OS
$\pi_1 \leftarrow$**First-Plan**(FP, Π_1)
$j \leftarrow 1$
Repeat until **Termination**
$\quad j \leftarrow j + 1$
$\quad$If $j > N$ Then $j \leftarrow 1$
$\quad \phi_{j-1}$ **Builds** augmented solution, $S_{j-1}^@$:
$\quad\quad$ • the plan $\pi_{j-1}^@$ and
$\quad\quad$ • the relevant information of problem $\Pi_{j-1}^@ = \{F_{j-1}^@, A_{j-1}^@, I_{j-1}^@, G_{j-1}^@\}$
$\quad \phi_{j-1}$ **Communicates** $S_{j-1}^@$ to agent ϕ_j
$\quad \phi_j$ creates a new planning task, ϕ_j':
$\quad\quad$ • it **Merges** its assigned problem Π_j and $S_{j-1}^@$
$\quad \pi_j \leftarrow$**Second-plan**(SP, Π_j)
If solved, return last plan

Fig. 4 High-level description of MAPR planning algorithm. In the second and following iterations, when $j = 1$, then $j - 1 = N$, where N is the number of agents on Φ'

plan heuristic detects a dead-end, then $c(g, \phi) = \infty$. Each agent communicates to a central agent each $c(g, \phi)$. The central agent defines a cost matrix, $c(PG, \Phi)$. Next, we have devised seven goals assignment strategies: `all-achievable`, `rest-achievable`, `best-cost`, `load-balance`, `contract-net`, `all`, and `subset`.

- `all-achievable`: assigns each goal g to all agents ϕ_i such that $c(g, \phi_i) < \infty$; that is, if the relaxed plan heuristic estimates g could be reached from the initial state of agent ϕ_i, then g is assigned to ϕ_i. So, it can assign the same public goal to more than one agent.
- `rest-achievable`: assigns goals iteratively. It first assigns to the first agent ϕ_1 all goals that it can reach (cost less than ∞). Then, it removes those goals from the goals set, and assigns to the second agent all goals that it can reach from the remaining set of goals. It continues until the goals set is empty. The agents order might be relevant. Thus, we have defined several orderings, as specified later.
- `best-cost`: assigns each goal g to the agent that can potentially achieve it with the least cost: $\arg\min_{\phi_i \in \Phi} c(g, \phi_i)$.

Table 1 Example of two agents, $\phi=\{R1,R2\}$ and three goals (humans using P1, P2 and P3)

	Goals		
Agents	(used H1 P1)	(used H3 P2)	(used H4 P3)
R1	7	10	12
R2	7	6	6

The table reflects the estimated cost of achieving each goal by each agent. It is computed as FF heuristic (length of the relaxed plan)

- `load-balance`: tries to keep a good work balance among agents. It first computes the average number of goals per agent, $k = \lceil \frac{|PG|}{m} \rceil$. Then, it starts assigning goals to agents as in `best-cost`. When it has assigned k goals to an agent, it stops assigning goals to that agent. The next goals that could be assigned to this agent will be redirected to the second best agent for each goal. At the end, agents will have either all k goals, or $m - 1$ agents will have k goals and one agent will have the remaining goals, $| PG | -k \times (m - 1)$.
- `contract-net`: it is inspired by the well-known negotiation scheme in multi-agent literature [62]. Under this setting, the virtual central agent takes the first public goal $g_1 \in PG$ and assigns it to the best bidding agent, where each bid is $c(g_1, \phi_i)$. Let us assume, it is ϕ_{b1}. Then, it selects the second goal g_2 and assigns it to the best bidding agent as before. However, in order to compute $c(g_2, \phi_{b1})$ it takes into account that g_1 has already been assigned to ϕ_{b1}, so ϕ_{b1} computes the estimated cost to achieve both g_1 and g_2, while the rest of agents only compute the cost of achieving g_2. Usually, but not necessarily, the cost of achieving both g_1 and g_2 for ϕ_{b1} is more expensive than using another agent ϕ_{b2} for achieving only g_2. In summary, at each iteration (over PG), `contract-net` assigns the current goal to the best agent, taking into account all previous assignments.
- `all`: assigns each goal g to all agents, independently of whether each agent can achieve the goal or not. It is a naïve strategy that allows MAPR and CMAP to use all agents, instead of using a subset of them.
- `subset`: as explained in the previous section, there are domains where agents must collaborate; e.g., one agent has to achieve one subgoal for another agent as moving a package to a given location. The `subset` goal assignment iterates over all goals. For each public goal $g_i \in G$, it computes the relaxed plan as in previous goal assignment strategies. However, instead of just computing the cost, it computes the subset of agents that appear in the relaxed plan for that goal, $\Phi_i \subseteq \Phi$. Those are agents that could potentially be needed to achieve the goal g_i. It is computed as the subset of agents that appear as arguments of any action in the relaxed plan. Then, it assigns g_i to all agents in Φ_i. A side effect of this strategy is that the set of selected agents for the combined planning task is the union of the agents subsets for all goals: $\Phi' = \bigcup_{g_i \in G} \Phi_i$.

Let us see how each strategy of goal assignment works in the example of the Depots-robots domain. The estimated cost of achieving each goal would be computed by each agent using the relaxed plan, generating Table 1. For instance, for R2 to achieve P2,[8] it first has to move P3 out of the way:

```
(take-right(R2,P3,C33,C43),move-left(R2,C43,C33)),
```

---

[8] For simplicity, we will refer to goals by the pod that has to be used by a human. In this case, achieving P2 means achieving (used H3 P2).

and then move to P2, take it and carry it to C31 where the human will use it. In the relaxed plan, R2 does not need to drop P3 before moving, since it did not delete (`free R2`) when it took P3.

The following is the set of goals that each strategy would assign to each agent. Given that there is a tie in the estimated cost of achieving P1, if needed we will assume R1 is selected for P1 given that it is the first agent in the problem. We discuss agent ordering techniques in the next subsection.

- `all-achievable`: R1(P1,P2,P3), R2(P1,P2,P3).
- `rest-achievable`: R1(P1,P2,P3), R2().
- `best-cost`: R1(P1), R2(P2,P3), due to tie breaking on P1.
- `load-balance`: R1(P1), R2(P2,P3), due to tie breaking on P1.
- `contract-net`: assigns P1 to the best agent. Since there is a tie, let us assume it will select R1. Then, it recomputes the estimated cost of each agent moving P2, given that now R1 also has to move P1. And assigns P2 to the best agent. The cost of moving P2 for R2 would be 6, while the cost of R1 moving both P1 and P2 would be 12 (7 for moving P1 and 5 for moving P2 afterward). So, it would assign P2 to R2. Now, `contract-net` recomputes the estimated cost of each agent moving P3, given that R1 has to move also P1, and R2 has to move also P2. The estimated cost for R1 would be 14. It would need 7 actions to achieve P1. Then, when going to take P3, it can start in C12, given that it passed through it and did not delete the fact that it was at C12. Thus, it would require 7 more actions to take P3 to H4. Similarly, the estimated cost for R2 would be 10. Thus, the final assignment would be: R1(P1), R2(P2,P3). As `load-balance`, it indirectly also tries to obtain a good balance.
- `all`: R1(P1,P2,P3), R2(P1,P2,P3). In general, the result would be different than the one of `all-achievable`; for instance, if there is a given agent that cannot achieve any goal, `all` would select it, while `all-achievable` would not.
- `subset`: computes the relaxed plan for achieving each goal and selects all agents in all relaxed plans. The relaxed plans of the goals would include R1 for P1 (due to tie breaking) and R2 for the other two pods, since the relaxed plan does not delete the initial position of R2 which is closer to all pods. Thus, `subset` would assign goals as: R1(P1), R2(P2,P3).

In configurations `rest-achievable`, `best-cost` and `contract-net`, there can be agents for which MAPR does not assign public goals. See, for instance, `rest-achievable` did not assign goals to R2 in the previous example. If they do not have private goals, those agents will not be used by MAPR for planning. The output of this step is a set of $N \leq m$ pairs (N agents assignments), where each pair is an agent $\phi_i \in \Phi$ and the set of public goals assigned to it, $G_i \subseteq PG$. The N selected agents (they have been assigned at least one public goal or have at least one private goal) compose the subset $\Phi' \subseteq \Phi$. Only agents in Φ' will be used in the next steps of the algorithm.

4.3 Ordering

The ordering of agents might be relevant for MAPR. We have defined four ordering schemes: `name`, agents are ordered by their given names in the problem description (in case the user has defined agents in a specific order this might be useful); `max-goals`, agents with more assigned goals are ordered first; `min-goals`, agents with less assigned goals are ordered first; and `random`, agents are randomly ordered.

Table 2 R1 plan for achieving the goal (used H1 P1). On the left, we show the unobfuscated plan, and on the right we show the actual plan generated by R1

(a) Unobfuscated plan	(b) Obfuscated plan generated by R1
move-right(R1,C14,C24)	anon3(C14,C24)
take-right(R1,P1,C24,C34)	anon7(P1,C24,C34)
move-left(R1,C34,C24)	anon4(C34,C24)
move-down(R1,C24,C23)	anon6(C24,C23)
move-down(R1,C23,C22)	anon6(C23,C22)
move-left(R1,C22,C12)	anon4(C22,C12)
move-down(R1,C12,C11)	anon6(C12,C11)
drop(R1,P1,C11)	anon11(P1,C11)
uses(H1,P1,C11)	uses(H1,P1,C11)

Agents ordering is relevant in two steps of MAPR: (1) before assigning goals; and (2) before iterating over the selected agents to generate subplans. In relation to (1), only name and random are used, since the other two ordering schemes depend on the assigned goals. In relation to (2), we can use all four schemes. In case there is a tie, such as assigning two agents the same number of goals, MAPR uses the name ordering.

4.4 Planning

Once goals have been assigned to the subset of N agents in Φ', planning starts by calling the first agent to solve its planning task. The task will be composed of its private planning task and its assigned public goals. If it does not solve the problem, it just passes the empty plan to the next agent. It could be either because there is no such plan, or because its plan needs some propositions to be achieved by the plans of the remaining agents. So, either the rest of agents solve the problem, or, eventually, it will be called again to solve the planning problem, but with some extra information coming from the other agents planning episodes. The following planning episodes can either use a planner or a replanner. In the latter case, apart from the domain and problem definitions, replanners take a previous solution as input [7,33].

Let us continue with the Depots-robots domain. Assume that we have used the load-balance goal assignment, leading to the assignment of P1 to R1 and P2 and P3 to R2. Thus, MAPR would first call R1 to generate a plan for achieving the goal (used H1 P1). The plan would be the one shown in Table 2 where we have shown the unobfuscated version on the left and the obfuscated one on the right. It is obfuscated given that agents use the obfuscated version generated at start. As we have seen, there are further obfuscation methods that are applied before sharing it with the following agents.

4.5 Building an augmented solution

If the first agent solves the problem, then it passes relevant information to the other agents. Since domains and problems are already obfuscated, in this step at each iteration the corresponding agent builds an augmented obfuscated solution to be used by the following agents. An augmented obfuscated solution is a solution found by any agent, augmented with domain and problem components needed for the other agents to reuse it.

A key issue for MAPR is what should one agent ϕ_j pass to the next agent ϕ_{j+1}. First, it has to pass the goals G_j it was assigned to achieve (both public and private), plus the goals of all previous agents that were passed previously to ϕ_j (let us call $\mathcal{G}_j$ the set of all these goals, including its own goals). Second, ϕ_{j+1} might not be able to generate a plan for those goals (because some of them might be private goals of previous agents), or it might find that the actions used by ϕ_j are preferred over the ones of ϕ_{j+1}. Therefore, ϕ_j also passes the instantiated actions' descriptions of the actions in the plan that ϕ_j used to achieve all the goals in $\mathcal{G}_j$. Given that we could use a second planner that is able to reuse the previous plan, it also passes the plan that ϕ_j used to achieve $\mathcal{G}_j$. Thus, a replanner might spend less time planning when reusing the previous plan.

Finally, ϕ_{j+1} will need the private part of ϕ_j's initial state (and the one of all the previous agents) to be able to regenerate ϕ_j plan. But, in order to preserve privacy by providing other agents as little information as possible, only the relevant part of that initial state is needed: those literals that are preconditions of actions in ϕ_j's plan. An alternative would be to regress over the goals to obtain the literals from the initial state that are really needed, discarding those that are added by some action in the plan, a_p, that another action needs, a_c, and are not deleted by some other action between the execution of a_p and a_c.

Therefore, an augmented obfuscated solution $S_j^@$ consists of the obtained plan and the set of components that are needed by the other agents to regenerate that solution if needed. More specifically, if agent ϕ_j generates the plan $\pi_j = (a_1, \ldots, a_t)$, it communicates $S_j^@ = \{\mathcal{A}_j^@, \pi_j^@, \mathcal{I}_j^@, \mathcal{G}_j^@\}$ to the next agent, ϕ_{j+1}. We explain next each component of $S_j^@$.

Actions $\mathcal{A}_j^@ = \{\text{PDDL}(a_i) \mid a_i \in \pi_j, \text{not original}(a_i)\}$, where $\text{PDDL}(a_i)$ is the instantiated (no variables) PDDL description of action a_i in the plan. There can be three kinds of actions in the plan: those that were shared by previous agents, $\mathcal{A}_{j-1}^@$; those whose name was obfuscated by ϕ_j given that they had a private predicate or a variable of a private type related to ϕ_j; and those that are the original ones (the ones whose definition has not been changed by any agent). Given that the ones of the third type are in the set of all agents, MAPR does not share those with the following agents. When ϕ_j obfuscates its private actions, it performs the second level of obfuscation explained in 3.4.

Plan $\pi_j^@ = \{a_1^@, \ldots, a_t^@\}$ is the obfuscated plan, where each $a_i^@$ is the result of applying the previous obfuscation steps. If macro-operators are learned, then π_j will be formed by either just one action, or a smaller set of actions than the original t actions.

Goals $\mathcal{G}_j^@ = \mathcal{G}_{j-1}^@ \cup G_j^@$ are all goals (private and public, including goals of previous agents) of agent ϕ_j.

Initial state $\mathcal{I}_j^@$ is the relevant initial state. Since MAPR only needs to pass to ϕ_{j+1} the relevant private part of the state, it only considers the literals that are preconditions of any action in the plan. Since it has already removed static predicates from actions preconditions, no private static literal from the initial state will be shared. Therefore, it is computed as:

$$\mathcal{I}_j^@ = \{f \mid f \in I_j^@, a_i \in \pi_j, f \in \text{pre}(a_i)\}$$

4.6 Communication

Each agent communicates $S_j^@$ to the next agent. We assume there is no noise in the communication. The size of the messages depends on: the plan size (number of actions in the generated plans π_j); the number of goals, where often $\mid \mathcal{G}_j \mid < \mid \pi_j \mid$; and the size of the initial state, $\mid \mathcal{I}_j \mid$. Thus, the size of messages is linear with respect to the plan size and initial state size. Since MAPR only communicates after each planning episode, the communication cost is proportional to the number of planning episodes and the size of each communication. If the

number of selected agents is $N = |\Phi'|$, the number of planning episodes is between N (the problem is solved in the first iteration after all agents have generated their plans) and performing k iterations until the problem is solved or resources are consumed (time or memory). So, communication cost is in the order of $O(kN(|\pi| + |G| + |I|))$. Furthermore, if MAPR learns only-one-macro-operator, then $|\pi| = 1$. This is a low communication overhead compared with other approaches that broadcast search decisions [54,68].

4.7 Merging

Each agent ϕ_{j+1} receives $S_j^@$ and builds a new planning problem by adding (performing the union of sets) the instantiated actions to its actions set, the goals to its own goals, the private previous initial state to its own initial state and all new propositions to its own propositions set. So:

$$\Pi_{j+1} = \{F'_{j+1}, A_{j+1} \cup \mathcal{A}_j^@, G_{j+1} \cup \mathcal{G}_j^@, I_{j+1} \cup \mathcal{I}_j^@\}$$

where $F'_{j+1} = F_{j+1} \cup \mathcal{G}_j^@ \cup \mathcal{I}_j^@ \cup L(\mathcal{A}_j^@)$, and $L(\mathcal{A}_j^@)$ are all the literals in preconditions and effects of actions in $\mathcal{A}_j^@$,

$$L(\mathcal{A}_j^@) = \{l \mid a_i \in \mathcal{A}_j^@, l \in (\text{pre}(a_i) \cup \text{eff}(a_i))\}$$

ϕ_{j+1} would now call the planner to generate a new plan that would achieve the goals in $\mathcal{G}_{j+1}$, which also takes into account ϕ_j goals. It can reuse parts of the plan of ϕ_j, or it could generate a plan from scratch that does not use ϕ_j, such as only using ϕ_{j+1} for all goals.

4.8 Termination

Given that each planning task incorporates all previous goals, including the private ones of the previous agents, as soon as the last agent finds a plan achieving all goals, the whole planning process finishes. If the last agent does not find a plan and there is still time, MAPR iterates over all agents again, but with the accumulation of goals. Starting in the second iteration, as soon as an agent finds a solution, then the whole planning task finishes, since it incorporates all goals from all agents. The planning process will terminate with failure only if the time or memory bounds are reached, or a maximum number of iterations are performed. We set the maximum number of iterations as five. We have found experimentally in the tested domains that MAPR only performs more than one iteration when the problem cannot be solved by MAPR.

4.9 Plans parallelization

MAPR generates totally ordered (sequential) plans. However, in MAP, plans are going to be executed by a set of agents. Therefore, parallel plans are preferred over sequential plans, so that other agents do not have to wait for executing their next actions when an agent is executing an action in the sequential plan. We have implemented an algorithm to transform a sequential plan into a parallel plan. First, a suboptimal algorithm generates a partially ordered plan from a totally ordered one by using a similar algorithm by Veloso et al. [73]. Then, a parallel plan is extracted from the partially ordered plan. The parallelization algorithm is planner independent. It receives two inputs: a planning task, Π, and a sequential plan, π, that solves the task. It outputs a parallel plan that is one of the potential parallelizations of the sequential plan.

4.10 Properties

First, as we mentioned before, MAPR performs *suboptimal planning*. Second, MAPR is *incomplete* mainly in two different scenarios: when there is at least one of the goals that needs more than one agent to achieve it and when there is a dead-end due to a strong interaction among agents. In the first case, consider, for instance, the Logistics domain. In almost all problems, more than one agent has to partially contribute to the achievement of single goals. For instance, in the standard *agentification* (definition of the sets AT, PP and PT) of the domain, trucks and airplanes are agents. If a package is initially at the post-office of a city `city1` and has to be delivered to the airport of another city, `city2`, we first need a truck to move the package from the source post-office to the airport of `city1` and then use an airplane to move it to `city2`. Thus, the goal of having the package at `city2` cannot be achieved only by any of these two agents in isolation. So, they return no solution when executed even if at least one solution exists. A potential solution to this problem could be to assign the same goal to both trucks and airplanes. However, this solution does not work without further changes in the algorithms. The truck in the first city would need to achieve a subgoal of the goal (that a package is in the airport of the first city, so that the airplane can actually achieve the goal). However, given the same domain, by just using a different agentification, such as airplanes as the only agents (as a real-world application for an airline company), MAPR can solve all IPC problems.

The second case of incompleteness can be observed in very tightly coupled domains. Suppose a domain with two agents A1 and A2, two resources R1 and R2 (both can only be used once), and two goals G1 and G2. A1 can use both resources, and can only achieve G1. A2 can only use R1 and can only achieve G2. The goal assignment strategy would assign G1 to A1 and G2 to A2. Suppose that the ordering scheme decides to start with A1 and it uses R1 to achieve G1. In A2's turn, A2 will fail, since it can only use R1 which has been used by A1 to achieve G1. It cannot make A1 use R2 instead (since A1 did not pass that action), and there is no way for A2 to generate a valid plan for both G1 and G2. When planning comes back to A1, it will also fail, for similar reasons. No IPC domain shows this kind of strong interaction.

Third, MAPR is *sound* if the first and second planners are. Intuitively, given that all goals (public and private) are propagated, if the last agent solves the problem (in the first iteration) or any agent solves the problem (in the next iterations), the plan must be applicable from the initial state of the propagated initial state and its application must result in a state that achieves all goals. Experimentally, we run the IPC[9] software [47], that automatically validates solutions. It validated all MAPR solutions.

Fourth, MAPR generates a *totally ordered plan*. But, we have seen how to convert a total-order plan into an equivalent parallel plan in case it is needed for a concurrent execution. The complexity of the first part of the conversion (from totally ordered to partially ordered) is quadratic in the number of actions in the plan. The complexity of the second part (from partially order to parallel plan) is again quadratic in the number of actions in the plan. In case macro-operators are shared among agents, there is no potential conflict among agents' plans (e.g., two robots occupying the same cell in depots-robots). When plans are sequential, actions of other robots have to wait until previous actions have been executed. The effects of each macro-operator summarize the changes in the state by all actions included in the macro-operator. When plans are parallelized, the parallelization code ensures there is no conflict among the parallelized actions (macro-operators in this case), so that no action deletes a literal

---

[9] International Planning Competition, ipc.icaps-conference.org.

that is needed by the other actions running in parallel (mutex actions cannot be executed in parallel.

Fifth, in relation to privacy preserving, the base method used by MAPR consists of a combination of: obfuscation by substitution, removal of private types and agents from literals, removal of objects from private predicates and removal of static predicates. Also, the actions and components of the initial state that are shared are a subset of the whole domain model and planning task. When an agent ϕ_j receives from another agent ϕ_i the augmented obfuscated solution, $S_i^@$, it receives a partial view of ϕ_i's knowledge. The privacy of ϕ_i will be assured if ϕ_j cannot gain knowledge about the private knowledge of ϕ_i. If we analyze separately the provided information in $S_i^@$, the private literals of ϕ_i have been converted into propositions. Since there is no relation in the initial state among literals (no common objects as arguments, as they are propositions), inferring the relation among them becomes harder. The same applies to private goals.

In relation to actions, we have described several techniques for increasing the level of privacy preservation. In real-world situations, static predicates contain much of the private data of agents, such as different operation costs, times, providers, or agents' preferences. So, MAPR can effectively preserve a great component of privacy after removing them. Another example of how this step highly improves privacy-preserving over simple obfuscation is in domains where agents traverse a network (e.g., roads, or connected depots). One truck (company) agent will not be able to infer the places another agent has visited in order to deliver all goods. It will only be able to infer the places where it had to pick-up and drop the goods (since those actions will require/modify the location of the goods). Thus, just removing static predicates gets rid of almost all semantics in some domains (road networks, costs, preferences, …). Also, if we add the learning of macro-operators, we remove further semantics of actions (they become compact representations of subplans). A minor negative side effect of macro-operators is that they tend to make MAPR slightly less efficient, as results of experiments show.

Finally, since some of the goal assignment strategies can reduce the number of agents to be used, each individual (agent) problem to be solved is much smaller than the original, even if we have $N \leq m$ such problems. STRIPS planning is *PSPACE*-complete [18]. Thus, both the original problem and each subproblem is still *PSPACE*-complete in the worst case. However, as the experimental results presented in the Sect. 6 show, by decomposing the problems into subproblems, solving all these N subproblems often takes less time than solving the original problem, even if we deal with privacy. This is specially true in the case of harder planning instances. In order to achieve the benefits, MAPR takes as input a domain-dependent characterization of agents and privacy that, as a side effect, allows MAPR to easily compute the decomposition.

5 Multi-agent planning in CMAP

In order to make our approach complete, we have devised a centralized variation of MAPR, named CMAP (Centralized Multi-Agent Planning). MAPR uses a decomposition scheme for MAP. Thus, as explained in the introduction, it takes into account two aspects of MAP: privacy and problem decomposition. An alternative way of using some of MAPR ideas consists of a centralized approach that also takes into account privacy.

Algorithm CMAP $(\Phi, M, PP, PT, GA, OS, P, \phi_{m+1})$: plan

---

Φ: a set of agents
M: MAP task
PP: list of private predicates
PT: list of private types
GA: goal assignment strategy
OS: agents ordering scheme
P: planner
ϕ_{m+1}: central agent

$PG \leftarrow$ ComputePublicGoals(M, PP, PT)
Order agents according to OS
Assign subset of PG to each agent ϕ_i using GA
Order agents according to OS
For each agent $\phi_i \in \Phi'$ (those that $G_i \neq \emptyset$) do
$\quad \phi_i$ **Communicates** $\Pi_i^@$ to agent ϕ_{m+1}
$\quad \phi_{m+1}$ merges $\Pi_i^@$ into the centralized planning task, Π_{m+1}:
$\quad\quad \Pi_{m+1} \leftarrow$**Merge**$(\Pi_{m+1}, \Pi_i^@)$
Return $\pi_{m+1} \leftarrow$**Plan**(P, Π_{m+1})

Fig. 5 High-level description of the CMAP planning algorithm

5.1 CMAP **algorithm**

The main difference with respect to MAPR is that CMAP first selects a subset of agents, then it combines all their obfuscated domains and problems into one single planning task and finally calls another centralized agent that performs a single planning episode with the combined problem. Figure 5 presents the algorithm. It takes as input the set of agents, Φ, the MAP task M, that was generated by the previously described MAP compilation. It also takes as input the list of private predicates and types (for computing the public goals), the goal assignment strategy, the ordering scheme, the planner to be used, and the virtual agent ϕ_{m+1} that will perform the planning step. We will comment next on two issues relative to CMAP: the planning episode; and its formal properties. The rest of steps are common to those of MAPR.

5.2 Obfuscation and planning

After the goal assignment, each selected agent sends its planning task (obfuscated domain and problem) to a centralized planning agent, ϕ_{m+1}. This agent merges all planning tasks and performs a centralized planning step with all the obfuscated information. Therefore, CMAP takes into account the privacy issue (by obfuscation) and benefits from the problem decomposition by using only a subset of the agents.

5.3 Properties

CMAP shares some of the properties with MAPR. It is *sound*, *suboptimal* and generates a *totally ordered plan* (that can be converted to a parallel one). It is suboptimal, since we are using a suboptimal planner (this could easily be fixed by using an optimal planner), and due to the

reduction on agents to be considered for planning (unless we use the `all` goal assignment). It is also *incomplete* for the second reason of sub-optimality: given the reduction of agents to be considered, there will be tasks where it is not able to solve problems. However, CMAP has the advantage over MAPR that it can be *complete* by using the `all` goal assignment strategy if a complete planner is used, and it can be made optimal if an optimal planner is used. We report experiments on completeness (by using CMAP and `all`), but we do not report experiments on optimal planning.

One of the added benefits from CMAP is that any state-of-the-art planner can be used to solve the compiled version, and we obtain the benefits of advances in the state-of-the-art in classical planning. Also, there is almost no communication cost (just the initial communication of the domain and problem of each agent to the central one). Its efficiency and previous good properties come at the cost of providing a weaker privacy-preserving behavior than MAPR, since it cannot use the second obfuscation, elimination of static private predicates, nor sharing of macro-operators.

The main assumption of CMAP in relation to privacy preservation is that we can protect privacy by obfuscation. Then, indirectly CMAP assumes that: either the centralized agent (the one performing the computation) can be trusted, and in that case we would have strong privacy-preserving properties; or, otherwise, the obfuscated versions of the domains and problems do not allow agents to practically infer knowledge about the private components of other agents (states and actions). In the second case, each agent is indirectly providing a finite state machine (FSM, through the states and actions) and other agents could infer part of the other agents' knowledge by matching the public components, and then, by applying some kind of backwards analysis from the known knowledge, infer the related knowledge on the corresponding FSMs.

6 Experiments and results

In this section, we describe the experiments we have performed to test our approach. We consider three sets of experiments: (1) comparing different parameter configurations, considering the two MAP algorithms, the goal assignment strategies and the ordering of agents schemes; (2) analyzing scalability of our approaches; and (3) comparing our work with similar work. We next explain the common parts of the experimental setups while each subsection describes the corresponding details. The section concludes with an analysis of the level of privacy achieved by MAPR.

6.1 Experimental setup description

We present here some common aspects of the experiments reported in the next subsections.

6.1.1 Metrics

Our objective in this paper is to improve planning efficiency in MAP tasks. We use the time-score metric of IPC'11. In particular, we use *time1* that computes the score of a planner p for a given planning task Π as:[10]

---

[10] http://www.plg.inf.uc3m.es/sw-ipc2011/.

$$\text{time1}(p, \Pi) = \begin{cases} \dfrac{1}{1+\log\left(\frac{T_{p,\Pi}}{T_{\Pi}^*}\right)} & \text{if planner solved the task} \\ 0 & \text{otherwise} \end{cases}$$

where T_{Π}^* is the minimum time required by any planner to solve problem Π, and $T_{p,\Pi}$ is the time it took planner p to solve Π. Any $T_{p,\Pi} < 1$ second is treated as one second.[11]

Although our main focus is planning efficiency, as quality of the solutions is a common measure in planning, we include some results on the length of the solution plans, as well as on the number of the parallel steps of the parallel plans (usually called makespan). The IPC assigns the following quality score to each configuration (planner) p and problem Π:

$$\text{quality}(p, \Pi) = \begin{cases} \dfrac{Q_{\Pi}^*}{Q_{p,\Pi}} & \text{if planner solved the task} \\ 0 & \text{otherwise} \end{cases}$$

where Q_{Π}^* is the best quality obtained by any configuration and $Q_{p,\Pi}$ is the quality obtained by p in problem Π. We use the same equation to report both plan length and makespan scores.

6.1.2 Planners

We used Fast-Downward code [39] to build a simplified version of LAMA- 2011, the winner of the sequential satisficing track of IPC'11.[12] We use it as the base planner of MAPR and CMAP and also as a centralized planner to compare with.[13] The sequential satisficing track aims at finding a solution to planning tasks that are deterministic and fully observable. LAMA- 2011 first runs a greedy best-first search with the FF and LM- COUNT heuristics and preferred operators, and unit cost (all actions are assumed to have unit cost). The goal of this first run is to find a plan as quickly as possible. Once a plan is found, it searches for progressively better solutions using a combination of greedy best-first and ωA^*. As the aim of our work is to study our MAP algorithms, focusing on planning efficiency, we have configured LAMA- 2011 to apply only the first search. We refer to this configuration as LAMA- FIRST during the whole section. LAMA- MK refers to executing LAMA- FIRST planner and then parallelizing the solution.

We have used LAMA- FIRST with unit costs for generating the plan for the first agent and LAMA- FIRST and LPG- ADAPT [33] for the successive planning episodes. While LPG- ADAPT is a replanning technique, LAMA- FIRST is not. We still call planning by reuse the configuration that uses LAMA- FIRST, because it can reuse the actions in the previous plans. LPG- ADAPT uses stochastic local search. We followed current practice (as in the IPC), running LPG- ADAPT only once per problem.

Our agents have been coded as function calls, so there is no overhead due to communication delays. Since the information that is being exchanged among agents is linear with respect to the size of plans and initial states, we do not expect a big overhead in planning time when transmitting it through a different communication channel. In any case, we have analyzed the time it takes each agent in MAPR to communicate its augmented solution to the next agent and it is always below 0.01 seconds.

---

[11] This avoids paying extra attention to differences in solving time under one second.

[12] http://www.plg.inf.uc3m.es/ipc2011-deterministic.

[13] We have made available the code of MAPR and CMAP at http://www.plg.inf.uc3m.es/~dborrajo/software/mapr-cmap-code.tgz.

6.1.3 Domains

We have used several IPC domains adapted for MAP that have been used in other MAP papers. Specifically, Elevators and Transport from IPC 2011; Rover, Zenotravel, Driverlog, Satellite, and Depots from IPC 2002 and Logistics, from IPC 2000.[14] We used the 20 problems defined in the corresponding IPC. We have used STRIPS models without action costs.[15]

The compilation of single-agent PDDL tasks (domain and problem) into MAP tasks requires three extra inputs, as we explained before. A.1 shows the inputs we have used to define the agents and the privacy level of each domain. In the first set of experiments, we have chosen agentifications that allow MAPR to solve some of the problems. For example, in the Logistics domain the agents are the airplanes. In case of considering the trucks as well, more than one agent is needed to achieve most of the individual goals and only CMAP with the subset and all goal assignment strategies can solve the problems. In the Elevators domain, there are two possible agent types: fast elevators for moving people quickly among blocks of floors; and slow elevators that stop at every floor of the block. A passenger usually needs a fast elevator to reach the required block, and then a slow elevator places him/her in the target floor. Thus, considering only the fast elevators as agents allows MAPR to solve most of the problems.

We have also defined two new domains: Depots-robots (the one we have used throughout the paper) and Port. The Depots-robots domain shows how our approaches work in a domain where all agents are able to achieve all goals. The Port domain deals with hoists that load crates into ships (Appendix A.2 provides the details of this domain). It is an example of the other extreme: how our approaches work in a domain where agents have only private goals (there are no public goals). Given that all goals are private, all goal assignment strategies perform the same assignment: a null assignment of public goals to each agent.

6.1.4 MAPR *and* CMAP *Configurations*

We will use: the two MAP algorithms, MAPR and CMAP; the seven goal assignment strategies (all-achievable (aa), rest-achievable (ra), load-balance (lb), best-cost (bc), contract-net (cn), all (all) and subset (sub)); and the four schemes for ordering the agents (name, min-goals (min), max-goals (max) and random (ran)). Agents ordering is performed before goal assignment, and at planning time. name and random are the only meaningful ordering schemes before goal assignment, since goals have not been assigned yet. At planning time, ordering agents is only meaningful for MAPR, since CMAP does not perform an ordered iteration over selected agents. So, MAPR can work with all ordering schemes, while only name and random make sense for CMAP. In the case of using CMAP with all and subset as the goal assignment strategies, the ordering schemes are irrelevant. all selects all agents, independently of their order. subset selects the union of the subsets of agents in the relaxed plans of all goals, and this is again independent of agents orderings. Besides, given the way that most problem generators work, the name and random strategies are equivalent, and thus, we do not report results on agents ordering for CMAP.

We name the systems as ⟨algorithm⟩-⟨goal assignment⟩-⟨ordering of agents⟩, omitting the last suffix (-⟨ordering of agents⟩) when the ordering scheme is name. mlpg refers to MAPR with LPG- ADAPT as a replanning system, while the name MAPR denotes that the

---

[14] The IPC Web page details the description of all the domains.

[15] In the following tables, names of domains have been shortened for space reasons.

replanning system is LAMA- FIRST. Finally, when using macros, `mapr-macros-oo` and `mapr-macros` refer to MAPR using macro-operators, sharing only-one-macro in the former and several macros in the latter.

6.1.5 Computational resources

We have used 1800 seconds as in the IPC. We performed most of the following experiments using the IPC'11 software [47]. Up to 6GB of RAM memory and 750GB of hard disk were available for each system. We run experiments in a cluster of Intel Xeon 2.93 Ghz Quad Core processor (64 bits) computers under Linux. The reported times include the whole process (unless specified); that is, the time taken to: (1) perform the MAP compilation, generating the domain and problem of each agent; (2) assign the public goals to agents; (3) find the sequential plan; and (4) generate the parallel plan. In order to better grasp where the solving time is spent, we have computed the average time spent in the used domains in these four steps. In the case of the IPC problems, each one of the steps 1, 2 and 4 takes less than one second in all domains and problems for most goal assignment strategies. Exceptions are described later. All solutions have been validated by using VAL [41] given that we use the IPC'11 software that includes the call to VAL.

6.2 Comparison among different configurations of MAPR and CMAP

In the first set of experiments the goal was to compare the different configurations of MAPR and CMAP.

6.2.1 Ordering the agents

First, we analyze the impact of ordering the agents in MAPR. Appendix A.3 shows the time-score metric using MAPR with the different goal assignment strategies and ordering schemes. For each goal assignment strategy, the differences among the results of each ordering scheme are not significantly large. So, we can conclude that the ordering of agents has little influence on the planning efficiency. The `min-goals` scheme is slightly better than the rest. Henceforth, unless specified, we use that ordering scheme in the remainder experiments. As we have discussed before, we do not report on agents' orderings of CMAP, since they are irrelevant for CMAP.

6.2.2 Goal assignment strategy

Next, we analyze the impact of the goal assignment strategies in CMAP and MAPR. We include also a comparison for reference with LAMA- FIRST (including a later step of parallelizing the solution). Table 3 shows the time-score results.

The `rest-achievable` strategy is mostly the best one, while `all-achievable` and `contract-net` are the worst ones, regardless of the used algorithm, MAPR or CMAP. This result underlines the correlation between planning efficiency and the number of agents involved in the planning process, due to the effect of problem decomposition. Appendix A.4 shows the number of selected agents by each goal assignment technique. In these domains, most of the agents can achieve by themselves all or almost all the goals. Therefore, the less number of selected agents, the better. Since `rest-achievable` is the strategy that usually selects a smaller set of agents, it is more efficient than the others. At the opposite end,

Table 3 Time-score metric using MAPR, CMAP and LAMA- FIRST, with the different goal assignment strategies

Planner	Logis	Rover	Satel	Driver	Zenot	Eleva	Depots	Port	Trans	d-rob	Total
mapr-ra	20.00	20.00	20.00	18.39	19.54	18.86	18.75	14.41	19.83	10.37	180.14
cmap-ra	20.00	20.00	20.00	19.07	19.85	18.69	18.84	13.15	19.65	8.74	177.99
mapr-bc	20.00	19.77	19.54	18.91	18.27	15.77	15.72	15.56	13.25	17.27	174.05
mapr-sub	20.00	19.77	19.54	18.55	18.22	16.82	14.47	14.01	13.16	11.97	166.50
cmap-sub	20.00	20.00	20.00	19.39	18.66	18.47	14.66	13.18	11.65	8.90	164.91
lama-mk	20.00	20.00	20.00	19.40	18.21	18.32	15.39	12.79	9.70	10.96	164.76
cmap-bc	20.00	20.00	20.00	19.27	18.31	16.98	16.19	12.40	7.34	9.12	159.61
cmap-all	20.00	19.77	20.00	19.02	18.22	17.78	14.72	12.67	6.73	10.02	158.94
mapr-lb	20.00	19.45	19.07	18.24	17.85	14.50	11.42	15.30	9.64	12.43	157.90
cmap-lb	20.00	20.00	20.00	19.18	18.12	16.15	14.34	12.74	6.84	9.97	157.34
cmap-aa	20.00	19.77	20.00	19.00	18.16	15.75	14.25	13.42	6.83	9.91	157.09
cmap-cn	20.00	19.45	18.66	18.45	17.24	10.62	14.28	13.76	6.51	9.62	148.57
mapr-aa	19.77	19.45	18.52	17.79	17.64	13.57	12.15	14.94	6.66	7.39	147.87
mapr-cn	19.54	19.16	17.68	17.88	17.08	10.21	10.84	15.61	9.19	10.10	147.28
Total	279.31	276.57	273.01	262.53	255.35	222.49	206.04	193.94	146.97	146.76	

Results are the sum of the scores of the 20 problems. Rows are sorted according to the total score

the `all-achievable` and `contract-net` strategies usually involve the participation of most agents, and they obtain worse performance. In the case of `all-achievable`, it selects all agents that can achieve at least one goal. In the case of `contract-net`, it tends to create a balance among agents, so as a side effect it tends to select most agents. `load-balance` also involves most agents, but `load-balance` is faster than `contract-net`, due to implementation issues.[16] These differences will be more clear in the scalability experiments. The `best-cost` strategy usually reduces the agents required to solve the planning task, as well, and it is also well placed in the ranking. As a side effect, some of our MAP privacy-preserving algorithms behave better by a huge margin than LAMA- FIRST that does not preserve privacy.

CMAP-`all` differs from LAMA- MK on the obfuscated process performed on the former. Thus, the input domain and problem to CMAP-`all` and LAMA- MK are different and the pre-processing step of Fast-Downward could generate different compilations into SAS+. That explains the variations on the scores of both systems.

The Port domain maintains a similar behavior among the different configurations, because goals are private and the goal assignment always returns the empty list of public goals assigned to each agent. The Depots-robots is a challenging domain. Most of the configurations solved less than 15 problems, where the harder problems are defined over a grid of 10×10. On the opposite extreme, all the configurations solved the 20 test problems in the Rover, Satellite, Zenotravel, Driverlog and Logistics domains. They also solved all the problems in the Elevators domains except for the `contract-net` approaches that could not solve one problem. All the configurations solved more than 16 problems in the Port and Depots domains, while the number of solved problems in the Transport domain ranged from 13 to 20. Appendix A.5 shows the number of solved problems of the different configurations.

---

[16] Among other issues, computation of estimated costs can be done in parallel for `load-balance`, while `contract-net` requires a sequential computation.

Table 4 Quality-score metric for the 20 test problems using MAPR, CMAP, LAMA- FIRST and LAMA- 2011, with different goal assignment strategies

planner	Logis	Rover	Eleva	Satel	Zenot	Driver	Port	Depots	Trans	d-rob	Total
lama-2011	19.75	19.92	19.88	19.75	19.85	20.00	18.94	17.81	16.96	11.82	184.68
mapr-bc	18.64	18.46	18.27	15.54	16.47	15.74	14.74	14.70	15.75	15.32	163.62
lama-mk	18.62	19.15	18.61	17.43	17.93	14.38	14.63	14.14	14.80	11.79	161.47
mapr-sub	18.95	19.02	18.38	15.86	16.92	15.38	14.95	13.56	15.49	11.83	160.33
mapr-ra	19.30	17.29	18.70	15.62	16.49	14.37	14.91	16.12	17.01	8.50	158.32
cmap-ra	19.34	17.37	18.69	16.38	16.22	14.68	14.42	16.12	16.96	7.59	157.75
cmap-aa	19.18	18.76	18.47	18.12	17.37	14.47	14.40	13.39	10.34	11.31	155.80
cmap-sub	19.10	18.96	18.65	17.30	17.48	14.35	14.40	13.07	13.60	8.73	155.62
cmap-all	19.05	18.63	18.47	18.04	17.37	14.58	14.81	13.39	10.34	10.49	155.15
cmap-lb	19.18	18.68	18.47	17.81	17.37	14.64	14.41	13.34	10.34	10.47	154.70
cmap-cn	19.18	18.63	17.37	18.04	17.37	14.58	14.42	13.39	10.34	10.47	153.79
cmap-bc	19.12	18.67	18.43	17.16	17.47	14.41	14.41	13.75	10.77	8.68	152.87
mapr-lb	18.68	18.33	18.32	15.63	14.57	15.19	15.27	11.30	13.45	11.70	152.44
mapr-aa	19.15	18.54	18.45	16.26	15.63	15.52	14.73	12.61	11.38	7.57	149.85
mapr-cn	18.64	18.86	17.48	15.71	15.21	13.99	14.73	10.63	14.09	9.35	148.70
Total	285.89	279.28	276.63	254.65	253.70	226.27	224.16	207.30	201.61	155.61	

Quality metric is plan length. Rows are sorted according to the total score

6.2.3 Quality

Even if the focus of this paper is not on improving quality of solutions, we report here on the solutions' quality obtained by MAPR and CMAP. Table 4 shows the quality-score metric when the quality metric is plan length. We also compare the results with LAMA- 2011 with the original code that participated in the IPC'11 as a reference. LAMA- 2011 improves solutions quality over time until it runs out of time, while our approaches stop as soon as one solution is found.

The score differences between LAMA- FIRST and the MAPR approaches are small. In fact, MAPR-`best-cost` outperforms LAMA- FIRST. This shows that our algorithms do not significantly penalize the quality of the plans. The quality metric gives a zero to the unsolved problems. In the domains where the score of MAPR-`best-cost` is higher than the score of LAMA- FIRST, the former solved more problems except in the case of the Driverlog where both systems solved the 20 problems. In theory, the solutions quality of `contract-net` should be better than the one of `load-balance`, since the estimation of costs is more precise. However, we see that in practice there is not much difference in the scores of these two configurations. The main reason is that `contract-net` approaches usually solve fewer problems, since `contract-net` takes more time to compute the goal assignment than `load-balance`.

We include now a study of the behavior of our algorithms in relation to the makespan of the generated plans, since most MAP approaches use makespan to measure plan quality. As we explained before, MAPR and CMAP generate sequential plans, but we can transform a sequential plan into a parallel plan. Table 5 shows the quality-score metric when the quality metric is makespan. LAMA- FIRST scores better than our approaches, so even if it is guided toward achieving goals as soon as possible, indirectly the generated sequen-

Table 5 Quality-score metric for the 20 test problems using MAPR, CMAP, and LAMA- FIRST, with different goal assignment strategies

planner	Logis	Eleva	Rover	Satel	Zenot	Driver	Port	Depots	Trans	d-rob	Total
lama-2011	18.85	18.94	18.90	17.63	19.21	19.13	17.42	17.74	18.61	11.46	177.89
lama-mk	16.08	17.27	15.94	15.86	17.41	12.89	13.37	13.95	15.47	10.08	148.32
cmap-aa	18.17	17.00	16.30	16.45	16.62	12.63	13.29	11.96	9.41	10.73	142.55
cmap-all	17.52	17.00	16.11	16.40	16.62	12.66	14.40	11.96	9.41	9.85	141.94
cmap-lb	18.17	17.00	15.90	17.00	16.62	12.73	13.00	11.92	9.41	9.70	141.45
cmap-cn	18.17	16.15	16.11	16.40	16.62	12.66	13.10	11.96	9.41	9.70	140.28
cmap-sub	17.45	17.19	15.82	15.39	15.91	12.47	13.29	10.12	11.21	8.40	137.25
cmap-bc	17.71	16.60	14.77	14.59	15.28	12.53	13.00	10.89	9.69	7.44	132.50
mapr-sub	16.28	16.41	16.27	13.90	12.70	12.80	12.68	9.51	9.43	9.34	129.32
mapr-bc	15.01	15.49	13.46	11.65	12.36	12.64	12.06	10.89	9.27	12.02	124.84
cmap-ra	15.91	16.74	12.86	13.25	12.31	11.99	13.10	11.32	9.27	4.96	121.71
mapr-lb	14.57	15.04	14.57	12.57	10.57	14.57	12.28	8.56	7.56	9.00	119.29
mapr-aa	15.71	15.58	15.55	13.33	12.67	13.12	11.46	9.23	6.06	6.46	119.17
mapr-cn	14.79	15.36	14.62	14.13	11.67	12.47	11.46	7.73	8.27	7.23	117.74
mapr-ra	16.30	16.85	12.18	11.93	11.57	10.46	11.48	11.32	9.28	5.30	116.66
Total	250.67	248.62	229.35	220.48	218.13	195.77	195.40	169.03	151.76	131.66	

Quality metric is makespan. Rows are sorted according to the total score

tial plans yield good makespans. Given that CMAP performs centralized planning, it has a comprehensive view of the whole planning process, and thus can more easily improve the quality of plans. Therefore, the CMAP configurations obtain better results than the MAPR ones. Among those, the goal assignment strategies that work best are the ones that try to obtain a good balance (`load-balance` and `contract-net`) and the ones that select more agents (`all-achievable` and `all`), indirectly allowing more agents to participate in the plan, and thus reducing the makespan.

6.3 Scalability study

In this section, we present a study on scaling up the difficulty of the problems in each domain, except for Transport, Depots-robots and Port. We did not use those domains in this study, given that the planning tasks were already difficult for the planners. We have generated harder problems than the ones used in the IPC; 20 random new problems in every domain. We have increased the number of agents and goals. In order to define the complexity of the new problems, we used a similar approach as the one used by the organizers of the IPC'11. For each domain, we increased the number of agents and goals until some configurations exhaust the computer resources (time or memory) when solving the problems. The difficulty of these harder problem instances depends on the domain. For example, in the Depots domain, some configurations have difficulties solving problems with 6 agents and 13 goals, while in the Rover domain more than 100 agents and 120 goals are needed to exhaust computer resources. The experimental setup is similar to the previous ones, but we have only tested the `min-goals` scheme for ordering agents. Appendix A.6 shows a summary of the characteristics of the tested problems. Tables 6 and 16 show the results of the experiments

Table 6 Time-score metric for the hard problems using MAPR, CMAP and LAMA- FIRST with the different goal assignment strategies

planner	Rover	Driverlog	Satellite	Zenotravel	Logistics	Elevators	Depots	Total
mapr-ra	15.34	18.87	20.00	20.00	18.87	1.86	18.95	113.89
cmap-ra	15.34	12.25	9.94	11.46	17.81	2.37	18.92	88.09
cmap-sub	14.76	13.99	10.78	10.76	19.01	14.74	4.02	88.05
lama-mk	19.97	14.34	9.63	10.40	13.18	14.56	0.77	82.85
mapr-bc	15.12	17.74	16.52	14.19	8.36	4.47	3.24	79.63
cmap-all	13.66	12.72	9.94	9.97	12.80	16.34	1.64	77.07
cmap-bc	14.95	14.45	7.73	10.56	12.26	3.76	9.08	72.77
mapr-sub	14.68	8.09	10.67	12.45	6.92	9.64	4.03	66.48
cmap-lb	13.75	12.20	7.79	10.36	10.15	5.43	1.45	61.13
cmap-aa	13.55	11.46	9.32	10.10	9.12	4.70	2.36	60.61
mapr-lb	11.75	7.03	10.27	11.15	0.00	2.39	2.20	44.78
mapr-aa	5.62	2.74	8.71	9.92	0.00	0.00	2.62	29.62
cmap-cn	4.28	3.96	7.65	1.36	0.00	0.00	2.70	19.95
mapr-cn	2.12	3.17	6.43	1.37	0.00	0.00	2.00	15.09
Total	174.87	153.00	145.38	144.06	128.47	80.25	73.97	

Rows are sorted according to the total score

on scalability. Table 6 shows the time score on these new problems and Table 16 shows the coverage.

The `rest-achievable` strategy remains being the fastest one by a large margin. However, MAPR scales much better than CMAP (there is a difference of more than 25 points between `mapr-ra` and `cmap-ra`). There is little variation between the ranking with respect to the goal assignment strategies. But the differences among the scores are bigger now than before. For example, the score of our best approach exceeds in 31 points LAMA- FIRST's (LAMA- MK) score. Hence, the differences on solving harder instances come from the fact that MAPR solves smaller problems at each iteration, while CMAP has to solve much bigger problems than the IPC ones. Much of the score differences come from unsolved problems. If we look at the worst configurations, those using `contract-net`, it is mainly due to the time it takes to perform goal assignment in these harder instances that increases a lot. So, it is better to use `load-balance` than `contract-net` if we are aiming at better load balance.

Appendix A.5 shows the number of solved problems of the different configurations, which depends on the domain. In the Zenotravel, Driverlog and Depots domains the `contract-net` strategy has scalability problems as described above. We have observed that the assignment of goals in these domains consumes almost all the available time. For the other strategies, it takes less than one second to assign the goals. As we can also see, MAPR is much faster than CMAP with `rest-achievable`: it solves approximately the same number of problems, but the time score is much higher.

To better understand the scalability of MAPR with the `rest-achievable` strategy when the number of agents increases, Fig. 6 reports the total time taken to solve the 20 hard problems on two representative domains: Driverlog and Rover. The x-axis shows the number of agents of each problem (Driverlog/Rover). In the case of the Rover domain, the total time remains practically equal as the number of agents increases. However, in the Driverlog domain, the
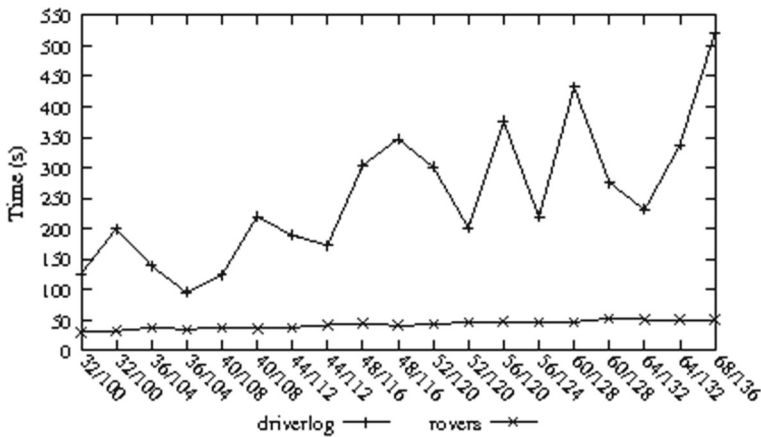
Fig. 6 Total time taken to solve the 20 hard problems of the Driverlog and Rover domains

time notably increases. The peaks of the graph are due to the fact that complexity of planning problems does not reside only on the number of agents.

6.4 Replanning algorithm

In this section, we want to study the impact of using a replanning system, MLPG (MAPR using LPG- ADAPT as second planner). So, we compare it against the configurations that obtained the best scores in the time-score metric using the hard problems of the above experiments and the problems of the challenging domains in the goal assignment experiments (Transport, Depots-robots and Port). We include in the comparison again the different orderings, since they have some impact as discussed below. Table 7 shows the results.

MAPR total score with the `rest-achievable` strategy is almost 52 points higher than MLPG scores. The `CMAP-rest-achievable` score is also much higher than MLPG scores. As we see, even with the advantage of reusing a past plan, MLPG performance is worse than the one of a *planning from scratch* system, as LAMA- FIRST. MLPG uses LAMA- FIRST for generating the plan for the first agent and LPG- ADAPT for the successive planning episodes, while MAPR always uses LAMA- FIRST. As a stand-alone planner, LAMA- FIRST is more efficient than LPG- ADAPT, due to all the improvements that it has over LPG (use of SAS^+, dual open lists, use of preferred operators, greedy best-first, . . .).

The only advantage of using LPG- ADAPT over LAMA- FIRST in the context of MAPR is that the successive planning episodes reuse the solutions communicated by the previous agent. That explains why `max-goals` is the best ordering scheme for MLPG; the first planning episode solves more goals, and the rest of planning episodes only have to add some additional plan steps for the new goals. Given how `max-goals` works, the second and later planning episodes will plan for smaller sets of goals than the one on the first step. Therefore, the previous plans can be mostly reused and planning is faster. When using a replanner the goal assignment strategy has little influence in the planning efficiency; all the strategies obtain very similar scores using the same ordering schemes. Let us take, for instance, `rest-achievable` and `all-achievable`. In `rest-achievable`, the second and later planning episodes will add a few new goals to the problem, so the planning tasks will be small. In `all-achievable`, the first episode (performed by LAMA- FIRST) will take care

Table 7 Time-score metric for the hard problems using MLPG and the different goal assignment strategies and ordering schemes

Planner	Satel	Trans	Depots	Zenot	Port	d-rob	Rover	Logis	Driver	Eleva	Total
mapr-ra-min	19.97	19.96	19.12	20.00	12.11	10.37	11.23	19.00	19.99	2.00	153.76
cmap-ra	9.93	19.79	19.09	11.46	11.75	8.72	11.23	17.93	13.02	2.96	125.88
mlpg-ra-max	17.98	14.42	14.68	13.58	10.59	9.22	19.80	1.47	0.00	0.63	102.37
mlpg-aa-max	18.16	14.34	14.54	13.62	11.25	10.45	19.80	0.00	0.00	0.00	102.15
mlpg-lb-max	17.95	14.28	14.66	13.58	11.08	10.44	19.82	0.00	0.00	0.00	101.81
mlpg-bc-max	18.07	14.32	14.99	13.57	9.27	10.44	19.80	0.00	0.00	0.00	100.45
mlpg-cn-max	18.03	14.53	14.37	13.55	9.85	9.58	19.81	0.71	0.00	0.00	100.43
mlpg-aa	18.30	14.68	14.03	13.72	13.67	10.23	6.92	0.00	0.00	0.00	91.55
mlpg-sub	18.31	14.47	13.39	13.83	13.16	10.40	6.93	0.00	0.53	0.00	91.01
mlpg-bc	18.25	14.70	13.84	13.80	12.62	10.45	6.94	0.00	0.00	0.00	90.60
mlpg-cn	18.28	14.23	13.86	13.73	12.08	10.45	6.94	0.00	0.54	0.00	90.11
mlpg-ra	18.26	13.92	13.65	13.71	12.72	10.41	6.92	0.00	0.00	0.00	89.61
mlpg-lb	18.32	14.61	13.89	13.77	10.67	10.22	6.95	0.00	0.00	0.00	88.43
mlpg-cn-ran	18.25	14.72	14.57	13.83	12.35	9.35	2.92	0.71	0.00	0.64	87.34
mlpg-lb-ran	18.26	14.65	13.89	13.76	12.19	10.43	2.93	0.00	0.00	0.64	86.76
mlpg-aa-min	18.74	14.48	15.10	13.82	13.22	9.50	0.00	1.39	0.00	0.00	86.25
mlpg-bc-min	18.71	14.25	14.86	13.64	15.16	9.11	0.00	0.00	0.00	0.00	85.73
mlpg-cn-min	18.79	14.65	14.96	13.75	12.70	9.56	0.00	0.00	0.00	0.00	84.41
mlpg-bc-ran	18.24	14.78	14.22	13.79	11.30	9.19	2.87	0.00	0.00	0.00	84.41
mlpg-lb-min	18.64	14.48	15.21	13.66	11.96	9.11	0.00	0.00	0.00	1.34	84.39
mlpg-ra-ran	17.86	14.86	14.40	13.49	10.56	9.35	2.92	0.00	0.00	0.64	84.08
mlpg-aa-ran	18.24	14.72	13.67	13.53	11.02	9.21	2.91	0.00	0.00	0.64	83.95
mlpg-ra-min	18.69	14.33	14.60	13.64	12.20	10.46	0.00	0.00	0.00	0.00	83.92
Total	414.22	344.18	339.61	318.84	273.48	226.65	177.66	41.21	34.08	9.48	

Rows are sorted according to the total score

of most goals (since we are dealing with domains where most agents can achieve most goals). And then the following episodes will add again very few goals. So, the expected behavior of `rest-achievable` and `all-achievable` is very similar when using replanning (LPG- ADAPT), while it is very different when using planning from scratch (LAMA- FIRST).

6.5 Comparison with state-of-the-art MAP

In this section, we compare against state-of-the-art MAP. We first present the comparison with a non privacy-preserving MAP approach, and then a comparison with several privacy-preserving MAP approaches.

6.5.1 Comparison with other non privacy-preserving MAP Approaches

Crosby et al. proposed the *Agent Decomposition-based Planner* (ADP) that automatically detects the agents in planning tasks and then performs an iterative search over the discovered agents [23]. As ADP does not preserve privacy, we first run our obfuscation method to convert

Table 8 Time and quality (makespan, Mk) scores and coverage (C) of PADP, MAPR-`rest-achievable-min-goals` and CMAP-`all`

Domain	PADP			CMAP			MAPR		
	Time	Mk	C	Time	Mk	C	Time	Mk	C
Satellite	20.00	15.44	20	20.00	18.99	20	20.00	14.71	20
Logistics	20.00	17.52	20	20.00	19.46	20	20.00	18.11	20
Rover	18.00	15.15	18	20.00	19.23	20	20.00	14.29	20
Driverlog	19.80	18.84	20	19.25	17.78	20	18.91	14.61	20
Zenotravel	16.65	16.13	18	18.47	19.02	20	20.00	13.19	20
Elevators	20.00	16.50	20	14.39	18.53	20	14.37	18.79	20
Depots	12.02	6.90	15	15.52	17.02	18	19.70	15.68	20
Transport	8.84	11.69	15	6.80	12.61	13	20.00	13.83	20
Depots-robots	13.00	12.60	13	9.44	9.55	11	10.21	5.65	11
Port	0.00	0.00	0	13.10	14.92	17	15.74	13.36	16
Total	148.31	130.77	159	156.97	**167.11**	179	**178.93**	142.22	**187**

the input domain and problem into the corresponding obfuscated versions. Then, we give those obfuscated versions to ADP as input, as we were doing with LAMA- FIRST in the previous experiments. We named it PADP (Privacy-preserving ADP). Thus, we convert a non privacy-preserving multi-agent planner (ADP) into a weak privacy-preserving one. We have compared it with MAPR and CMAP. We used the same IPC domains we have used previously, as well as the two new domains Port and Depots-robots with their 20 problems. Table 8 shows the results of the comparison.

The total scores of MAPR and CMAP are higher than the scores of PADP. PADP did not solve a significant number of problems; four of them were not solved because they are problems with a single agent and the algorithm returns a null output. This shows that ADP is better than LAMA- FIRST in some domains and that we can help a system that does not preserve privacy (such as ADP) to include some level of privacy, through the first obfuscation method. However, it cannot benefit from stronger privacy preservation techniques as in the case of the ones used by MAPR.

6.5.2 Comparison with privacy-preserving MAP Approaches

Finally, we compare MAPR (the best version found so far: `rest-achievable-min-goals`) and CMAP (`all`) against FMAP [66], MAFS [54], and MADLA [27], since they represent the current state-of-the-art on suboptimal privacy-preserving MAP. We do not provide comparisons with other techniques because they are way behind in coverage (number of solved problems), time score, or because they solve a different planning task (such as optimal approaches, or planning with self-interested agents). In the FMAP paper, its authors showed that FMAP outperforms by a great margin their previous approach, MAP- POP [68].[17] PLANNINGFIRST [56] could only solve the first two problems in the Rovers domain and the first problem in the Satellite domain. Other approaches, as the one presented by Jonsson & Rovatsos [42], also show running times well above the ones we will present here in the satisficing version. They also provide results for the optimal case, but we do not deal with optimal planning in this

---

[17] We already reported similar results in a previous paper [5].

paper. Also, they take as input a plan. We cannot compare with (MA- A*) [55] either, since their approach performs optimal planning. In the Related work section, we provide a more comprehensive list of approaches and their differences with ours.

Domains We have used the same domains as in the previous sections, which are most of the ones used by the authors of FMAP [66]. We used the same instances they used. Since in some domains they used more than 20 problems, we selected the first 20 problems of their benchmark, so that all domains have the same weight in the scores.[18] We could not compare in some domains, given that they have a different input definition than our approaches. As an example, in Openstacks, they used agents that are not in the IPC domain nor problem definitions.

FMAP takes as input a domain for each type of agent and a set of problem descriptions, one for each agent. So, when running FMAP, we have used their agentification of the domain. On the other hand, approaches based on MA- STRIPS, such as MAFS and MADLA, fail to solve any problem in agentifications of domains where there is at least one action with no agent in their parameters (for instance an agentification of the Logistics domain with airplanes as only agents, or Driverlog with drivers as agents). Thus, in order to test different agentifications and their impact in the behavior of planners, we have used two alternative agentifications for some domains (see details in Appendix A.1). Domains with a suffix *-f* in the name mean that the systems used the agentification proposed by FMAP to solve the problems. The differences with our agentification affect only the private predicates in the Satellite, Rover, Driverlog, Transport and Zenotravel domains. In the Logistics, Depots and Elevators domains, differences lie on who the agents are. MADLA uses untyped PDDL definitions. Thus, we report its results only on the domains provided by its authors.

Other variables setup Goal assignment. We have used our best configuration for MAPR (with `rest-achievable`). Also, we have compared with the complete version: CMAP-all. *Ordering of agents.* We have used the best configuration for MAPR (`min-goals`). Ordering of agents is irrelevant when using CMAP-all, since it uses all agents. *Planners.* We have used LAMA- FIRST with unit costs for generating the plan for the first and consecutive agents. We also compared obfuscation by learning only-one-macro-operator (MAPR- MACROS- OO) or learning several of them (MAPR- MACROS). *Time bound.* We have used 1800 seconds as in the IPC. We used a different computer for this experiment: a 2.6GHz Intel Core i7 with 4Gb of RAM running MacOS X.[19] *Metric.* We compare here the approaches with respect to time and makespan.

Table 9 shows the results of the time score. It does not show the totals, since it good be unfair for systems that cannot handle all different agentifications. So, the analysis has to be performed over domains or agentifications of domains. We highlight in bold the best configuration per domain and agentification.

As it can be seen, MAPR and CMAP obtain a huge difference in time scores with respect to the rest of state-of-the-art approaches. The best approach of the rest is MAFS, followed by FMAP and MADLA. The difference in score is higher in domains where agents alternate private and public actions in their plans, because MAFS and FMAP have to continuously communicate the changes in the states. They are usually tightly coupled domains. This can be seen for instance in Port, Depots-f, and Transport. As expected, some approaches get penalized using some agentifications: MAPR in Depots-f, Logistics-f and Elevators-f; and MAFS in Driverlog (both versions), Depots, Logistics, Elevators and Depots-robots. Others

---

[18] This decision favors them since FMAP scales worse than MAPR and CMAP and the more difficult problems are the last ones.

[19] FMAP needs a specific execution setup that was difficult to reproduce in the cluster.

Table 9 Time score of FMAP, MAFS, MADLA, CMAP-all and MAPR-rest-achievable-min-goals with and without macros

Domain	CMAP	MAPR	MAPR-macros	MAPR-macros-oo	FMAP	MAFS	MADLA
Satellite	19.59	**19.63**	18.68	17.54		16.99	
Satellite-f	**19.75**	19.64	18.71	17.49	8.58	15.77	10.54
Rover	**19.80**	19.78	18.97	18.11		15.00	
Rover-f	**19.83**	19.79	18.66	18.13	10.13	14.36	11.26
Driverlog	**19.85**	18.89	18.25	18.17			
Driverlog-f	**19.82**	19.20	18.79	18.33	9.13		1.00
Zenotravel	18.29	**19.98**	19.73	19.35		15.85	
Zenotravel-f	18.26	**19.98**	19.70	19.77	11.88	15.10	
Transport	8.56	19.32	19.67	**19.86**		0.9	
transport-f	8.37	19.63	**19.99**	19.49		1.66	
Depots	16.15	**19.28**	19.12	18.36			
Depots-f	**18.00**				2.11	7.98	1.03
Logistics	**20.00**	**20.00**	**20.00**	**20.00**			
Logistics-f	**20.00**				3.95	19.01	1.05
Elevators	18.34	**19.76**	19.66	19.20			
Elevators-f	**20.00**	1.00	1.00	1.00	11.91	**20.00**	0.0
Port	15.05	16.66	**17.18**	3.29		7.84	
Depots-robots	9.61	**9.80**	9.61	9.76			

We show in bold the best configuration per domain

simply cannot solve problems in a given domain due to complexity: FMAP in Transport-f; and MADLA in Elevators-f.

Differences within the MAPR versions are small. Therefore, even if using macro-operators provides smaller scores, they improve on privacy preservation. So, we can balance privacy and efficiency, depending on the desired levels for a particular domain. Also, CMAP and MAPR scores are quite similar in the domains that both can handle. CMAP on the other hand does not suffer from different agentifications, but it has a weaker privacy preservation.

Table 10 shows the results of the quality score measured as makespan. MAFS and MADLA only report plan length values, so we report the length as the makespan of their plans. Thus, their scores are low. In the case of MAFS, it does not return a plan, so we could not run our parallelization algorithm. In the case of MADLA, the plans are returned obfuscated, so they would have to be first deobfuscated to be parallelized. Therefore, the only fair comparison is against FMAP. The focus of our techniques has not been yet on improving quality. However, given that we solve many more problems, the quality score is better than that of the others. Also, even in the cases where FMAP, MAPR and CMAP solve the same problem instance, manual inspection of results show that our quality is usually not that far from that of FMAP and sometimes even better (FMAP is suboptimal with respect to makespan). Only in the Elevators domain FMAP scores higher than CMAP.

As with respect to modeling the MAP domains, we use the IPC version of the domains, so we have very little modeling overhead; just the one needed to provide the values for the agent types (AT), private predicates (PP) and private types (PT). FMAP use some multi-agent-oriented representation of the domains with most predicates represented as functions. These versions are semantically equivalent to the ones of the IPC, even if they include more

Table 10 Quality score (makespan) of FMAP, MAFS, MADLA, CMAP-all and MAPR-rest-achievable-min-goals with and without macros

Domain	CMAP	MAPR	MAPR-macros	MAPR-macros-oo	FMAP	MAFS	MADLA
Satellite	**19.48**	15.05	15.59	15.66		10.65	
Satellite-f	**17.82**	14.03	14.68	14.68	15.46	9.68	3.14
Rover	**19.60**	15.26	15.97	15.9		6.14	
Rover-f	**19.02**	14.38	14.96	15.01	16.15	5.73	3.87
Driverlog	**19.61**	15.19	14.98	15.04			
Driverlog-f	**17.58**	13.53	13.31	13.42	14.65		1.00
Zenotravel	**19.78**	14.21	14.21	14.14		8.41	
Zenotravel-f	**18.26**	12.94	12.94	12.87	16.32	7.34	
Transport	**16.00**	13.12	13.14	13.05		0.11	
Transport-f	**16.00**	13.10	13.05	13.06		2.40	
Depots	**17.53**	16.17	16.20	15.14			
Depots-f	**18.00**				4.94	6.20	0.20
Logistics	**19.89**	18.25	18.25	18.25			
Logistics-f	**19.44**				8.29	5.88	0.02
Elevators	**19.16**	18.65	18.27	18.95			
Elevators-f	16.93	0.50	0.50	0.50	**18.53**	10.14	
Port	**17.03**	15.86	16.86	3.24		5.70	
Depots-robots	**10.92**	6.79	6.79	6.79			

information related to some multi-agent aspects. So, our approaches can deal with a simplified version of privacy (slightly richer than the one of MA- STRIPS), while FMAP can use richer semantics.

6.6 CoDMAP results

We participated in the First Competition of Distributed and Multiagent Planners (CoDMAP) with the following versions of our systems:

– CMAP- Q: CMAP algorithm with the subset goal assignment strategy and LAMA- 2011 as the base planner (including its anytime behavior). It aims at optimizing plans' quality and coverage.
– CMAP- T: CMAP algorithm with the subset goal assignment strategy and LAMA- FIRST as the base planner. It aims at optimizing planning time and coverage.
– MAPR- P: MAPR algorithm with the min-goals scheme for sorting agents, LAMA- FIRST as the base planner and learning only-one-macro. It aims at maximizing privacy among agents.

They placed as follows in the centralized track:[20]

– CMAP- Q: **1st place - IPC quality score**, 7th place - coverage score, 12th place - IPC time agile score
– CMAP- T: **2nd place - IPC time agile score**, 4th place - coverage score, 5th place - IPC quality score

---

[20] See detailed results at http://agents.fel.cvut.cz/codmap/results.

– MAPR- P: 7th place - IPC time agile score, 11th place - IPC quality score, 12th place - coverage score, 12th place

However, as organizers stressed in the presentation of results, it could be considered more a *comparison* than a *competition* given that, among other issues, privacy preservation behavior greatly varied among competing planners. For instance, the first two planners in coverage were based on ADP, so they did not preserve privacy. The third one, based on SIW [48], had a very light privacy-related scheme, so agents had full access to other agents states. So, CMAP- T was the first planner to preserve an equivalent level of privacy as other MAP planners (such as the ones presented earlier).

6.7 Analysis of privacy preservation

As already stated, obfuscating agents' private information only ensures weak privacy. Guaranteeing strong privacy in MAP is hard to achieve. However, it is feasible to verify experimentally privacy by extracting the information exchanged among the agents in the planning process and then establishing whether an agent is able to infer private data of the other agents, as it was done in [10] for some domains.

CMAP assumes that either: the centralized agent can be trusted, and thus CMAP would show strong privacy-preserving properties; or, otherwise, the obfuscated versions of the domains and problems can only ensure weak privacy.

Regarding MAPR, various parameters influence privacy preservation: the goal assignment strategy, the agents ordering scheme, the use of macro-operators, the domain model, the elements considered private in the domain (defined by PT and PP), and the agentification (defined by AT). We will study next the most relevant aspects of the relation of these parameters and privacy.

The goal assignment strategy determines the set of agents involved in the planning process. Hence, the excluded agents do not exchange any private information with the selected agents, since they do not participate in the planning process. Then, goal assignment strategies that select less agents in average tend to provide stronger privacy preservation. In average, if we order goal assignment strategies according to this aspect from stronger to weaker privacy preservation we would have: `rest-achievable`, `best-cost`, `subset`, `load-balance`, `contract-net`, `all-achievable`, and `all`. A second impact on privacy preservation relates to the number of goals assigned by each strategy to each selected agent. If a strategy assigns many goals to each agent, the agents' subplans will be longer in average and the amount of obfuscated private data exchanged among agents will be bigger than with the other strategies. According to this criterium, the weakest goal assignment strategies in relation to privacy preservation would be `all-achievable`, and `all`.

The ordering of agents when they generate subplans also influences privacy. The first agent generates its partial plan without previous information, so it learns nothing from the rest of agents unless it is invoked again (this happens very few times in practice). The second agent receives the augmented obfuscated solution from the first agent. Thus, it only can infer private information from the first agent. The following agents receive all previous agents' solutions together, so they cannot distinguish which agent each obfuscated data belongs to. This is precisely one of the main differences with other approaches that run agent planners in parallel, given that each agent knows which other agent is sending the information when they change something from the public part of the state. While, in our case, each agent does not know which information belongs to which other agent.

Each individual agent can potentially infer private knowledge only from previous agents. Therefore, if we have an ordering of priorities of agents in privacy preservation, we can use

it to order the agents. In that case, the agent with the less restrictions on privacy preservation would be ordered first. The second agent could be the one with which the first agent does not mind losing some privacy preservation. And the following agents can be ordered from the less restrictive to the more restrictive ones in terms of privacy preservation.

The relation between the goal assignment strategy and the ordering of agents is also relevant in relation to preserving privacy. For instance, in our case, we use `min-goals` as agents' ordering scheme. Thus, it will order first the agent with less number of goals. If we combine it with `rest-achievable`, it will most probably have few goals to achieve. Thus, the weaker link (between the first and second agents) would only have weak privacy preservation for a small number of actions in average (the ones that achieve those few goals).

The macro-operators remove information about intermediate states, increasing the privacy level. So, we have analyzed the information exchanged by MAPR-`all-achievable` with macro-operators (when it learns only one) and without them to determine the private data the second agent and the following agents can infer from it. We have performed this analysis per domain, using some of the configurations studied in the experiments. Table 11 shows the results. The first columns display the private information the second agent could infer from the first agent when MAPR uses or not macro-operators. The last column displays the private information the following agents could reliably distinguish from previous agents. The table displays the private predicates we used to define the privacy level of the IPC domains used in the experiments. An underlined predicate means that the displayed agent may not be able to infer the corresponding private information.

Next, we explain the private data that may not be inferred per domain.

- Rovers: the goals are that a rover communicates soil, rock and image data located in public locations. The private data are (1) the rover's instruments, (2) the rover position, (3) the waypoints the rover can traverse, (4) the instruments, cameras and stores the rover has and its current state (full, empty, calibrate, available) and (5) the supported modes of its cameras and its calibration target. The rover equipment for soil/rock analysis and imaging can be inferred only in the case its assigned goals include to communicate a soil/rok/image data. But the number of instruments, cameras and stores the rover has remains private. The actions for moving and calibrating a rover include only private predicates with a public object. The second level of obfuscation removes these public objects. Then, if a rover has to navigate through a maze of waypoints, the other rovers will be able only to infer the first and last waypoints the rover traversed, given that they will appear in the macro-operator (while the other traversed waypoints will not appear in the macro-operator). Also, other agents will not be able to infer the rover calibration target if it is an intermediate waypoint in the traversed path.
- Satellite: the goals are that a satellite takes images of public directions with a public mode. The private data is: (1) the on-board instruments, the modes they support and their states (power-on and calibrated), (2) the calibration direction of the instruments and (3) the satellite direction. It is possible to infer that the satellite has an instrument that supports a mode, but the exact number of instruments and their states cannot be inferred. The action for calibrating a satellite includes only private predicates with a public object. The second level of obfuscation removes that public object from the action, and the macro-operator removes all information on intermediate propositions before each take-image action. So, other agents will not be able to infer the satellite calibration direction.
- Driverlog: the goals are that a package, a driver or a truck ends in a public position. The private data are the location of the driver, in a public location or in a public truck (`driving`). The predicate `driving` encodes intermediate states that the macro-operators remove.

Table 11 Analysis of MAPR privacy level in the IPC domains

Domain	Second agent		Rest of agents
	Without macros	With macros	
Rover (rovers)	at can_traverse	at can_traverse	at can_traverse
	equipped_for_soil_analysis	equipped_for_soil_analysis	equipped_for_soil_analysis
	Equipped_for_rock_analysis	equipped_for_rock_analysis	equipped_for_rock_analysis
	Equipped_for_imaging	equipped_for_imaging	equipped_for_imaging
	Empty have_rock_analysis	empty have_rock_analysis	empty have_rock_analysis
	Full have_soil_analysis	full have_soil_analysis	full have_soil_analysis
	Calibrated supports	calibrated supports	calibrated supports
	Available on_board	available on_board	available on_board
	Have_image store_of	have_image store_of	have_image store_of
	Calibration_target	calibration_target	calibration_target
Satellite (satellites)	Calibration_target	calibration_target	calibration_target
	Pointing power_avail	pointing power_avail	pointing power_avail
	Calibrated supports	supports calibrated	supports calibrated
	power_on on_board	power_on on_board	power_on on_board
Driverlog (drivers)	At-driver driving	at-driver driving	at-driver driving
Transport (trucks)	At-truck in capacity	at-truck in capacity	at-truck in capacity
Zenotravel (aircrafts)	At-airplane fuel-level In	at-airplane fuel-level in	At-airplane fuel-level in
Depots (trucks)	In at-truck	in at-truck	in at-truck
Logistics (airplanes)	Inside-airplane At-airplane	inside-airplane at-airplane	inside-airplane at-airplane
Elevators (fast-elevators)	Boarded-fast lift-at-fast reachable-floor-fast Passengers-fast can-hold-fast	boarded-fast lift-at-fast reachable-floor-fast passengers-fast can-hold-fast	boarded-fast lift-at-fast reachable-floor-fast passengers-fast can-hold-fast
Port (hoists)	Lifting assigned available At-ship on-ship	lifting assigned available at-ship on-ship	lifting assigned available at-ship on-ship
D-robots (robots)	At-robot carries free	at-robot carries free	at-robot carries free

Underlined predicates represent the private data the displayed agent may not be able to infer from previous agents (it may remain private for the previous agents). The analyzed agentification is displayed in parentheses below the domain name

Hence, the fact that a driver is driving a truck may remain private. Also, the final position of the driver could also remain private, given that it can walk somewhere else after leaving the truck in its final destination, and the preconditions and effects of the walk action are private and will not appear in the macro-operator.

– Transport, Zenotravel, Depots and Logistics: in these domains the agents are vehicles (trucks or airplanes) and the private data is their location, the internal properties as their capacity or fuel-level and the number of packages/persons they are transporting (`in` and `inside` predicates). The second level of obfuscation removes the public objects capacity and fuel-level, from the respective private predicates. So, the internal properties of the vehicles may not be inferred in most problems. In addition, MAPR increases the privacy level if it uses macro-operators, since they also remove the intermediate states and compact several actions into one.

– Depots-robot: the private data is the location of the robot and whether the robot is carrying a pod or it is free. The public predicate empty-cell allows all private data to be inferred. Macro-operators can only hinder inferring the details of the robots movements when they are loaded.

– Port: this domain deals with hoists that load crates into ships. Both, ships and hoists (the agents) are private together with the information concerning them, i.e., (1) the surfaces, crates and hoists a ship contains and (2) the current state of a hoist (whether it is available and it is lifting a crate). Each hoist is associated to a ship, so MAPR only maintains weak privacy, as it would happen with any other MAP planner. Once a hoist picks up a crate (public due to the position of the crate within the dock) and deposits it into its ship, the crate disappears (its position becomes private). Thus, any agent can infer that the crate is going to end up in the ship associated to the hoist.

– Elevators: this domain represents a set of elevators for moving up and down persons from one floor to another. The private data is the elevator position, the floors the elevator can reach, the number of persons it can hold and the persons it transports. The actions for moving up and down a elevator contain only private or static predicates. The second obfuscation removes the public objects from the private predicates and the static predicates. Hence, it becomes more difficult for other agents to infer the reachable floors of an elevator. Also, if we use macro-operators, the current occupation of the elevators and the number of persons it can hold would also be more difficult to infer.

6.8 Summary of results

As a summary, we can draw some conclusions:

Ordering of agents All the schemes behave similarly, being the `min-goals` scheme slightly better than the rest. Therefore, the ordering of agents has little influence on the planning efficiency.

Goal assignment The `rest-achievable` alternative is the fastest one, while `all-achievable` and `contract-net` are the slowest ones. We have also shown that in average `rest-achievable` uses much less number of agents. In the scalability study, `rest-achievable` continues to be the fastest one.

Centralized versus distributed algorithm In simple instances, given the same goal assignment strategy, MAPR is slightly faster than CMAP in almost all strategies. Also, when dealing with harder instances, MAPR-`rest-achievable` scales better, and outperforms by a great margin all other configurations.

Comparison with LAMA- FIRST There is a difference of more than 15 points between the time score of our best setting (MAPR-`rest-achievable`) and the LAMA- FIRST score (no

privacy) in the simpler problems. The difference is even greater in the case of harder instances between MAPR-rest-achievable and LAMA- FIRST (around 31 points).

Quality of solutions Our algorithms do not significantly penalize plans' quality. Even MAPR-best-cost scores slightly better than LAMA- MK in plan length. As expected, there is room for improvement in quality, given that our configurations do not focus on it, and we are not using all the available time for improving the first solution, as LAMA- 2011.

Replanning versus planning from scratch in MAPR Using LAMA- FIRST (planning from scratch) is better than using LPG- ADAPT (replanner). Therefore, there is room for improvement on better replanners. The difference between the scores of the configurations that use LPG- ADAPT are small, though. In these cases, the goal assignment strategy has little influence on the planning efficiency.

Comparison with state-of-the-art similar planners We have shown that both CMAP and MAPR outperform by a big margin state-of-the-art planners in the same setting that we use: deterministic suboptimal privacy-preserving STRIPS multi-agent planning. In the case of MAFS, its behavior depends on the interaction between public and private actions when each agent is searching for a solution. If each agent has to interleave often between public and private actions, then MAFS cannot benefit of the effects of tunneling. Thus, it will have to exchange often states and actions with the rest of agents, thus reducing its efficiency. This can be observed in domains as Port.

Using macro-operators By sharing macro-operators between agents in MAPR, we can obtain better privacy preservation at a slight decrease of efficiency.

7 Related work

MAP has been approached from both the multi-agent and the automated planning communities [24]. In general, most works in the multi-agent community have not used standard automated planning techniques, focusing on some related aspects as self-interested agents, cooperation, negotiation or scheduling [8,26,46]. On the other hand, most previous works in the planning community defined agents as resources and used centralized planning to solve MAP tasks. In some works, each agent generates a separate plan, and then those plans are merged [28,32,50,52]. These previous approaches differ from our work in that they focus on maximizing utility, or do not preserve privacy of goals or actions. Recently, the PSM planner and their variants implement a new algorithm for merging plans from different agents in the MAP setting with privacy. The initial plans also contain projections of public actions of other agents [69]. The main difference with MAPR and CMAP is that it focuses on the merging algorithm and it uses the MA- STRIPS model.

Some of the ideas in MAPR were already present at a high level in GPGP [46]. Particularly, distributed problem solving, exchange of partial plans and plan repair. But, if we analyze in more detail, there are strong differences with the planning mechanisms used in both. GPGP uses a goal hierarchy for planning, similarly to the work on HTN planning, very different from the planning models used in PDDL. They also deal with scheduling, and execution coordination, while we only deal with planning. Also, their joint intentions (goals) are only handled implicitly. So, agents could solve their planning tasks without taking into account a known jointly shared goal. In MAPR, all agents know about the public goals and cooperatively reason to achieve them.

Recently, there has been a renewed interest on developing MAP techniques for cooperative agents that explicitly consider the agents private information in the suboptimal [56,66,68] and

optimal settings [54,55]. Among the recent deterministic distributed planning approaches, we find the ones that use iterative plan refinement [42], distributed CSP [12,56], distributed A* [54,55], SAT [25], or partial-order planning [66,68]. Also, there is some interest on including mechanism design principles for self-interested agents [55].

Jonsson & Rovatsos described a MAP approach that is based on an iterative refinement process of successive single-agent planning episodes [42]. They start the planning iterations using an arbitrary initial plan in the joint-actions space. Then, they perform successive single-agent cost-optimal planning steps to obtain better plans. We perform a similar iterative single-agent process, but MAPR differs in that we do not need an initial plan, and it does not work in the joint-actions space; our plans are totally ordered, but we can later efficiently generate a parallel plan. Also, at each iteration, we perform suboptimal planning.

Another approach, μ-SAT, uses SAT planning for generating individual agents' plans and combining the solutions [25]. It focuses on two agents problems, while we are not restricted to the number of agents in the problem. In order to solve problems in domains as Logistics, they expand the Strips representation of actions with a new type of preconditions, external preconditions. Agents do not use those preconditions when planning, and they assume some other agent will make those preconditions true when needed. The focus of their paper is on optimal planning (minimizing makespan), even if in their experiments they only report on suboptimal planning. Also, in the experiments they assume each agent is able to apply all actions, and thus achieve all goals. Finally, they do not handle privacy.

Nissim et al. implemented ideas published in previous papers on using distributed CSPs to solve the planning task [56]. As we have discussed previously, the approach is theoretically sound, but in practice it is inefficient compared to other approaches. A later work compared this system with two other optimal MAP systems corroborating its inefficiency in practice [27]. Some of its authors have recently developed a distributed algorithm [54], that can be configured to perform optimal planning, MAD-A*, optimal planning for self-interested agents [55], or satisficing planning, MAFS.

MAP-POP [68] is based on agents that share their public information when performing partial-order planning. They also used an iterative refinement process that is able to work on both loosely coupled and tightly coupled domains. The same authors have developed a variation that is complete, FMAP [66].

A difference with some of these previous approaches is that they generate either partial, parallel or joint-actions space plans. In the case of MAPR and CMAP, plans are totally ordered (sequential) plans. This is not a real drawback of our approach. Computing the optimal partially ordered plan (and then the corresponding parallel plan) from a totally ordered plan is NP-hard in general [19]. However, as we have discussed, there are efficient suboptimal algorithms that can be used [73]. Another difference is that some of the other approaches are able to share incomplete plans or states, while MAPR in its current version is only able to share complete plans. Thus, MAPR cannot solve problems given some agentifications of the domains, as Logistics or Elevators, where agents cannot achieve goals just by themselves. As we have shown in the section on experiments, other approaches (such as those based on MA-STRIPS, MAFS, MADLA) find similar problems with other agentifications. As a solution, we have developed CMAP that can solve problems in those domains, given that it performs centralized planning. Finally, a key difference with all these previous approaches is that we use state-of-the-art single-agent planners. Therefore, we automatically benefit from advances in deterministic STRIPS planning. On the other hand, they cannot automatically incorporate new developments in planning into their MAP approaches.

The main difference between our method for protecting privacy and other distributed approaches [54,66] is that we are sharing an obfuscated action model as well as the private

initial state and goals. As we have discussed before, by using obfuscation by random substitution, removal of static predicates and macro-operators, we are providing the same amount of information as the one that MAFS or FMAP agents have.

In this paper, we have compared several task allocation strategies. Other decision-theoretic approaches use more complex strategies, as creating agents coalitions, or maximizing the utility [60,77]. In our work, we do not deal with coalitions, nor with utility maximization. We do use contract-net [62], selecting the lowest cost bid at each iteration, as a sealed-bid auction. In the context of decision theory, is very common to use a Vickrey [74] auction [55,71]. The first work considers a dynamic planning-auction mechanism, where each agent plans one goal at a time, and auctions those goals for which it cannot find a plan for. In relation to our approach, they perform dynamic goal allocation, and use plan repair when they receive the next goal, but the plan repair only deals with each agent's plan. Thus, the agents do not consider potential positive or negative interactions with the plans of the rest.

Another key topic we covered in this paper is privacy-preserving planning. Alternative approaches deal with the problem of MAP and the fact that agents do not know the other agents' knowledge as planning under partial observability [2,76]. The second paper poses the planning task using epistemic logic, instead of using belief states to represent the uncertainty on other agents' knowledge. Recently, Kominis and Geffner [43] have shown that generating Bolander and Andersen's multi-agent plans can be compiled into classical planning tasks. They allow their agents to sense the beliefs of other agents, so again it is a different planning task than the one with are dealing with here.

A related question is how much privacy we are loosing by sharing obfuscated augmented plans among agents. One alternative would be to use the approach proposed by van der Krogt [70], where he proposes to measure the loss of privacy using Shannon information theory. They propose to take into account the number of plans that could potentially be generated, and the number of plans that each agent observed. Others also use information theory to measure privacy loss in the context of CSP [76]. They also deal with the issue of privacy-preserving problem solving as problem solving under partial observability, given that private information of other agents can be dealt with as such. Another alternative to measure privacy loss is to consider it in the context of game theory [72].

One of the reasons MAPR is efficient comes from the decomposition of the problem into agent-based subproblems (it is not the case of CMAP). There have been other decomposition techniques in the literature that have been applied to planning both in the deterministic [11] and non-deterministic settings [38]. We work on the deterministic setting and the decomposition is induced by the definition of agents, and their private knowledge, instead of using other structural reasons to decompose variables and/or actions. Early approaches provided a manual agents' decomposition [45]. Recently, Crosby et al. proposed how to automatically detect agents in planning tasks and thus decompose the tasks into agent-oriented subtasks [23]. We believe this is an important step for automatically determining agents and their properties to decompose the planning tasks. Therefore, this work focuses on one of the aspects of MAP, problem decomposition, but leaves out the other, preserving privacy, which is fundamental for some applications. In most real-world applications with privacy concerns, privacy does not emerge from the structure of the domain and problem, but what the users define as private. Nevertheless, we have experimentally compared our approaches against their approach by providing them an initial obfuscated domain.

In related fields, as path finding, there is also a current strong interest on multi-agent research [59,63]. Approaches in this field also use centralized and distributed approaches, and are also divided in optimal and suboptimal solutions. A similar approach to MAPR is CA* [61], where a path is computed for the first agent, and then the positions that the agent has to

visit and the times when it has to visit them are reserved in a table. Then, the algorithm plans for the next agent, but taking into account that it cannot visit the same positions at the same time as the ones reserved by the first agent. In the case of MAPR, we allow the second agent to change the plan for the first agent, as far as its private and public goals are still achieved. The main differences of multi-agent path finding with our work is that they do not usually handle privacy, and we deal with domain-independent planning.

Another interesting application of MAP is on plot generation [16]. The work is based on Continual Multi-Agent Planning work by Brenner & Nebel [17]. They explicitly handle sensing, communication as well as physical actions of agents, that are modeled using the Multiagent Planning Language (MAPL). Their approach also allows to explicitly reason about agents (self and others) beliefs.

Finally, while we focus on deterministic MAP, there has been plenty of work on solving non-deterministic MAP tasks [14,58,65]. The advantage of these works over MAPR is that they can deal with uncertainty. But, usually, they do not scale up as well as deterministic MAP.

8 Conclusions

This paper presents two MAP approaches for cooperative agents with private information, MAPR and CMAP, as well as several methods to preserve privacy. In these two approaches, agents share public and relevant private information, preserving agents' privacy. Then, they perform distributed iterative planning (MAPR), and centralized planning (CMAP). Instead of performing parallel distributed searches as most work in MAP, MAPR and CMAP execute sequential problem solving. The second main difference with respect to previous work is the use of goal assignment in MAPR and CMAP. We have shown how different goal assignment strategies lead to very different performance of both planners in terms of time to solve or quality. These strategies encode different ways of performing division of labor among agents according to varying criteria. This is a key difference with respect to other MAP approaches that make a stronger focus on agents collaborating to achieve each goal.

In relation to the input description of MAP tasks, we hope that the MAP community defines in the future a standard for MAP tasks. In the meantime, other works have advocated for specific extended languages, thus changing the PDDL descriptions [15,44,68], or defining privacy as emerging from action descriptions [12]. Instead, we have preferred to receive the inputs in standard PDDL and let the user define privacy with three simple lists: agents types, private predicates and private types. From a knowledge engineering perspective, we believe these three lists are quite easy to define (it took us very little time for each analyzed domain).

MAPR calls each agent with the plans, goals and state literals from previous agents in order to regenerate previous solution (obfuscated) if needed, while reasoning at the same time on how to achieve the current agent goals as well as the other agents goals (private or public). We remove from the domain actions the parameters related to the agents type, so the search space diminishes. Actions in previous plans are added to the domain. So, there is a decrease in the number of instantiated actions that each agent has to deal with proportional to the actions that include the agents as a parameter. There is an increase in the number of instantiated actions equal to the sum of actions of the plan of the previous agent. In general, the search space of each agent is greatly reduced since the amount of new instantiated actions is usually much less than the amount of removed instantiated actions.

In relation to modeling, *agentification*, each system makes assumptions on agentification and solvability. MAP approaches based on MA- STRIPS (MAFS, MADLA) cannot solve any problem when there are actions where none of its parameters is an agent. Examples are: Logistics (trucks), Depots (trucks), Driverlog (driver). On the other hand, MAPR can solve problems in some MAPR-incomplete domains by changing the agentification, such as in the Logistics (airplanes) or Elevators (fast elevator) domains. On the other hand, CMAP completeness does not depend on the agentification.

Another key issue in MAP is preservation of privacy. We have defined three main techniques for obfuscating agents' knowledge (random substitution, macro-operators, and generation of zero-arity predicates). And we have implemented them, with some variations, such as removal of static predicates or different ways to learn macro-operators. We have shown that the level of privacy preservation in MAPR is similar to the one provided by other approaches (FMAP or MAFS). Instead, CMAP needs either a trusted central agent.

In the experiments, we have compared seven different strategies for assigning public goals to agents, as well as four different schemas for ordering agents. If we consider the IPC problems, we have seen that there are some approaches that have very similar performance in terms of efficiency (number of solved problems as well as time to solve), such as MAPR and CMAP with the `rest-achievable` strategy for assigning goals. Results also show that ordering agents has little impact on planning efficiency, being the `min-goals` scheme slightly better than the rest. Experimental results also show that MAPR and CMAP greatly improve the efficiency on planning time over state-of-the-art MAP approaches, while providing a similar level of privacy preservation (MAPR).

We have also used two different replanning alternatives for MAPR: a deterministic greedy best-first approach (LAMA- FIRST) and a stochastic local search replanning algorithm (LPG- ADAPT). It is remarkable that many MAPR and CMAP configurations obtain a higher performance than using a centralized approach that does not preserve privacy (LAMA- FIRST). In part, this is due to the fact that a secondary effect of distributing public goals among agents is that we reduce the number of agents that are used for planning (those for which MAPR or CMAP assign goals), effectively reducing the search space. Also, in the case of MAPR, each subproblem only considers the search space of the corresponding agent plus a much smaller search space of previous agents; the one that corresponds to the actions that were needed to achieve the previous agents goals. On the other hand, LPG- ADAPT configurations perform worse than the ones that use LAMA- FIRST. This does not imply that planning from scratch is better than replanning for this setting. In fact, other papers have shown the opposite. Further work remains to be done to show the benefits of using replanning with respect to planning from scratch in this setting.

Considering quality, the IPC score has a strong bias toward coverage. Therefore, the best configurations for quality are usually the ones that solve more problems; i.e., the ones using `rest-achievable` as the goal assignment strategy. While MAPR generates better solutions in terms of plan length, CMAP improves the makespan. In this paper, we are only computing the first solution. In case we would be interested on obtaining good quality solutions, we would have to run further searches to improve the solutions, as in LAMA- 2011.

As we have discussed, MAPR does not work in domains and agentifications where two agents have to collaborate to achieve a single goal. In those domains, there are at least two alternatives: changing the agentification as we have shown in the paper; or to use CMAP (`all`). We are currently working on other ways to handle that problem. A possibility consists on computing goal regression from each goal and assigning each agent a set of subgoals that it should address apart from the initial goals of the problem.

In the future, we would like to provide a more in-depth analysis of the main theoretical properties related to privacy preservation. It is still a open issue in the community. In some of the papers on MAP, there are some hints on how privacy can be preserved [10], but there is no formal framework yet that provides theoretical properties or methods to measure it. New developments in cryptography as the approach proposed by Gentry [35] describe fully homomorphic encryption schemes that allow an agent to manipulate data of another agent without actually being able to see the data itself. In our case, this scheme would allow each agent ϕ_i to pass other agents an encrypted function (or set of functions) that manipulate its private states and actions, without the rest of agents knowing (or being able to infer) the private data of agent ϕ_i.

Another line of research is on self-interested agents, by using the expected utility for each agent and goal as in our preliminary work [75]. Also, we would like to provide solutions to domains with joint (interacting) actions where an action uses two or more agents, such as two robots moving a table, as has been recently studied by Brafman & Zoran [9]. Planning for multiple agents shares some aspects with previous work that analyze symmetries among resources [49]. We would like to study the connection with that work and perhaps benefit from an analysis of the domain to detect unnecessary agents. Automatically inferring the agent-related information from the domains could also benefit modeling, as proposed by Crosby *et al.* [23]. Given that a planner (LAMA- FIRST) is better than a replanning system (LPG- ADAPT) for the reuse phase of MAPR, we pretend to work on better replanning systems. Finally, we would like to define new domains that focus more specifically on MAP tasks.

A Complement to experiments and results

A.1 Agentification of domains

Table 12 shows the inputs we have used to define the agents and the privacy level of each domain. The column *AT* represents the agents domain types, the column *PT* the private types and the column *PP* the private predicates. Some domains require small changes in the PDDL predicates to distinguish the private predicates from the public ones. For example, in the Driverlog domain, the predicate (at ?obj - locatable ?loc - location) is split into three predicates *at-truck, at-driver* and *at*, to represent the location of the trucks, drivers or packages, respectively. The table displays the transformed domains in italic font. The table also includes an alternative agentification of some domains. As planner FMAP uses it, we add the suffix *-f* to distinguish it from our agentification.

Table 12 Agents types (*AT*), private types (*PT*) and private predicates (*PP*) used to define the agents and the privacy level of the IPC domains used in these experiments

Domain	*AT*	*PT*	*PP*
Rover	Rover	Camera Store	At can_traverse equipped_for_soil_analysis Equipped_for_rock_analysis equipped_for_imaging Empty have_rock_analysis have_soil_analysis full Calibrated supports available have_image Store_of on_board calibration_target
Rover-f	Rover	Camera Store	At can_traverse equipped_for_soil_analysis Equipped_for_rock_analysis equipped_for_imaging Empty have_rock_analysis have_soil_analysis full Calibrated supports available have_image Store_of on_board calibration_target
Satellite	Satellite	Instrument	Supports calibration_target on_board Pointing power_avail calibrated power_on
Satellite-f	Satellite	Instrument	Supports calibration_target on_board Power_avail calibrated power_on
Driverlog	Driver		At-driver driving
Driverlog-f	dRiver		
Transport	Vehicle		At-truck in capacity
Transport-f	Vehicle		At-truck capacity
Zenotravel	Aircraft		At-airplane fuel-level in
Zenotravel-f	Aircraft		Fuel-level
Depots	Truck		In at-truck
Depots-f	Truck depot distributor		
Logistics	Airplane		Inside-airplane at-airplane
Logistics-f	Truck airplane		At at-airplane
Elevators	Fast-elevator		boarded-fast lift-at-fast reachable-floor-fast Passengers-fast can-hold-fast
Elevators-f	slow-elevator Fast-elevator		Lift-at reachable-floor passengers can-hold
Port	hoist	Ship	Lifting assigned available on-ship at-ship
Depots-robots	Robot		At-robot carries free

In italics the domains where some predicates were changed

Table 13 Time-score metric using MAPR and the different goal assignment strategies and ordering schemes. Each total is the sum of the scores of the 10 domains and the 20 problems per domain

Goal assignment	Agents ordering			
	`min-goals`	`random`	`name`	`max-goals`
aa	107.5	108.3	108.4	106.9
ra	116.5	116.0	116.6	114.6
bc	115.5	111.2	109.2	107.0
cn	111.4	109.9	108.4	108.4
lb	110.9	110.6	107.6	108.2
Total	561.8	556.0	550.2	545.1

A.2 The Port domain

The Port domain is similar to the *container stacking problem* [57], which is inspired by a real-world port and it is similar to the DWR domain described in the book [37].[21] In this domain, there is an area in the dock of a port with towers of containers with height of at most k containers (we used $k = 3$ in the experiments) placed in a grid of $n \times m$. In comparison with the Blocksworld domain there are fixed positions in the table, several arms and tables, and a maximum height of towers. The port has a set of hoists, one for each ship that is waiting to be loaded with containers. Initially, all containers are in the dock, and the goals are to load all of them in the ships; the goals establish to which ship each container should go and on top of which other container. In this domain, we have defined hoists to be the agents, and all the goals to be private. Thus, each hoist has a set of containers that it has to load in its corresponding ship, creating towers also specified in the goals. Goals derive from the `on-ship` predicate.

Also, this domain is quite tightly coupled, given that there is a high interaction among private goals of agents. In order for a hoist to load a container into a ship, it might have to move around the containers in the dock (the container might be the bottom container in a tower) and those containers are the ones that have to be loaded in other ships. So, there are both positive and negative interactions.

A.3 Ordering the agents

Table 13 shows the time-score metric using MAPR with the different goal assignment strategies and ordering schemes. It displays the total time-score considering the ten domains (Transport, Rover, Zenotravel, Driverlog, Satellite, Depots, Logistics, Elevators, Port and Depots-robots) and 20 problems per domain.

A.4 Selected agents

In order to see the effect on the number of selected agents by each goal assignment technique, Table 14 shows the average and standard deviation (over the 20 problems) of the number of agents that each goal assignment strategy requires for solving a problem. The first column shows the average and the standard deviation of the number of goals of the 20 problems,

---

[21] The domain definition as well as the used problem files can be found at http://www.plg.inf.uc3m.es/~dborrajo/software/mapr-cmap-domains.tgz.

Table 14 Average and standard deviation of the number of goals of the 20 test problems and of the number of agents used by each goal assignment strategy to solve the problems

Domain	Goals	All	aa	cn	lb
Rovers	10.0 ($\pm$ 4.5)	3.8 ($\pm$ 1.8)	3.8 ($\pm$ 1.8)	3.8 ($\pm$ 1.8)	3.6 ($\pm$ 1.8)
Satellite	16.4 ($\pm$ 9.7)	4.8 ($\pm$ 2.7)	4.8 ($\pm$ 2.7)	4.8 ($\pm$ 2.7)	4.6 ($\pm$ 2.5)
Zenotravel	12.0 ($\pm$ 7.6)	3.3 ($\pm$ 1.4)	3.3 ($\pm$ 1.4)	3.3 ($\pm$ 1.4)	3.3 ($\pm$ 1.4)
Depots	10.2 ($\pm$ 3.9)	2.9 ($\pm$ 1.4)	2.9 ($\pm$ 1.4)	2.9 ($\pm$ 1.4)	2.8 ($\pm$ 1.2)
Driverlog	14.0 ($\pm$ 8.7)	3.3 ($\pm$ 1.6)	3.3 ($\pm$ 1.6)	3.3 ($\pm$ 1.6)	2.5 ($\pm$ 1.3)
Transport	20.3 ($\pm$ 2.9)	4.0 ($\pm$ 0.0)	4.0 ($\pm$ 0.0)	4.0 ($\pm$ 0.0)	4.0 ($\pm$ 0.0)
Logistics	14.5 ($\pm$ 5.9)	1.6 ($\pm$ 0.5)	1.6 ($\pm$ 0.5)	1.6 ($\pm$ 0.5)	1.6 ($\pm$ 0.5)
Elevators	37.6 ($\pm$ 12.7)	3.0 ($\pm$ 1.0)	3.0 ($\pm$ 1.0)	3.0 ($\pm$ 1.0)	3.0 ($\pm$ 1.0)
Depots-robots	16.0 ($\pm$ 9.2)	3.8 ($\pm$ 1.5)	3.8 ($\pm$ 1.5)	3.8 ($\pm$ 1.5)	3.8 ($\pm$ 1.5)
Port	21.5 ($\pm$ 11.7)	10.4 ($\pm$ 6.7)	7.6 ($\pm$ 4.6)	7.6 ($\pm$ 4.6)	7.6 ($\pm$ 4.6)

Domain	bc	sub	ra
Rovers	2.7 ($\pm$ 1.2)	2.8 ($\pm$ 1.3)	2.2 ($\pm$ 0.9)
Satellite	3.3 ($\pm$ 1.9)	3.6 ($\pm$ 1.9)	2.7 ($\pm$ 1.5)
Zenotravel	2.5 ($\pm$ 1.4)	2.5 ($\pm$ 1.3)	1.6 ($\pm$ 0.9)
Depots	2.0 ($\pm$ 1.0)	2.2 ($\pm$ 1.2)	1.0 ($\pm$ 0.0)
Driverlog	3.3 ($\pm$ 1.6)	2.8 ($\pm$ 1.4)	2.8 ($\pm$ 1.4)
Transport	3.3 ($\pm$ 0.8)	2.8 ($\pm$ 1.3)	1.0 ($\pm$ 0.0)
Logistics	1.5 ($\pm$ 0.5)	1.5 ($\pm$ 0.5)	1.0 ($\pm$ 0.0)
Elevators	2.4 ($\pm$ 1.0)	2.1 ($\pm$ 1.1)	1.0 ($\pm$ 0.0)
Depots-robots	3.1 ($\pm$ 1.1)	3.6 ($\pm$ 1.4)	1.0 ($\pm$ 0.0)
Port	7.6 ($\pm$ 4.6)	7.6 ($\pm$ 4.6)	7.6 ($\pm$ 4.6)

and the remaining columns show both values for the number of agents for each strategy. As a baseline, strategy `all` requires all the agents in the problem. We can confirm that `rest-achievable` is the strategy that uses less number of agents in average. The standard deviations of the `rest-achievable` strategy decrease with respect to the baseline strategy, so the agents' reduction occurs in most of the problems. In these domains, most of the agents can achieve by themselves all or almost all the goals. Therefore, in domains as Transport or Depots, the `rest-achievable` strategy can even select only one agent for all problems. In Rovers and Satellite, it selects more agents given that both rovers and satellites have different instruments and might be in unconnected waypoints (rovers). The `best-cost` strategy is able to also reduce the number of selected agents, with the added advantage that selects the agents that can supposedly achieve the goals with least cost. `load-balance` and `contract-net` tend to select all the agents to achieve the desired load balance.

A.5 Coverage results

Table 15 shows the number of solved problems of the different configurations for the problems used to analyze the impact of the goal assignment strategies. And Table 16 shows the number of solved problems of the different configurations for the hard problems.

Table 15 Number of problems solved using MAPR, CMAP, and LAMA- FIRST with the different goal assignment strategies and the `min-goals` ordering scheme

Planner	Rover	Satel	Zenot	Drive	Logis	Eleva	Depots	Port	Trans	d-rob	Total
mapr-bc	20	20	20	20	20	20	19	17	20	18	194
mapr-ra	20	20	20	20	20	20	20	16	20	11	187
lama-mk	20	20	20	20	20	20	18	17	19	13	187
cmap-ra	20	20	20	20	20	20	20	17	20	10	187
mapr-sub	20	20	20	20	20	20	18	16	19	14	187
mapr-lb	20	20	20	20	20	20	16	17	17	15	185
mapr-cn	20	20	20	20	20	19	15	17	17	12	180
cmap-aa	20	20	20	20	20	20	17	17	13	13	180
cmap-sub	20	20	20	20	20	20	16	17	17	10	180
cmap-all	20	20	20	20	20	20	17	17	13	12	179
cmap-lb	20	20	20	20	20	20	17	17	13	12	179
mapr-aa	20	20	20	20	20	20	18	17	14	9	178
cmap-bc	20	20	20	20	20	20	17	17	14	10	178
cmap-cn	20	20	20	20	20	19	17	17	13	12	178
Total	280	280	280	280	280	278	245	236	229	171	

Table 16 Number of hard problems solved using MAPR, CMAP, and LAMA- FIRST with the different goal assignment strategies and the `min-goals` ordering scheme

Planner	Zenotravel	Rover	Satellite	Driverlog	Logistics	Depots	Elevators	Total
mapr-ra	20	20	20	20	19	20	2	121
cmap-sub	20	20	18	18	20	7	17	120
cmap-ra	20	20	15	17	18	20	3	113
cmap-all	20	20	18	17	16	2	19	112
lama-mk	20	20	17	18	16	2	17	110
cmap-bc	20	20	13	18	16	14	5	106
mapr-bc	20	20	20	20	11	6	6	103
mapr-sub	20	20	16	12	8	8	13	97
cmap-aa	20	20	17	16	13	3	6	95
cmap-lb	20	20	14	17	14	2	7	94
mapr-lb	20	17	17	11	0	4	4	73
mapr-aa	20	9	17	5	0	6	0	57
cmap-cn	4	8	18	8	0	4	0	42
mapr-cn	4	4	15	6	0	4	0	33
Total	248	238	235	203	151	102	99	

A.6 Scalability study

Table 17 shows a summary of the characteristics of the tested problems. Particularly, the range (minimum and maximum number) of agents and goals present among the problems of each domain.

Table 17 Minimum–maximum number of agents and goals of the 20 test problems

	Transport	Driverlog	Zenotravel	Rover	Satellite	Depots	Logistics	Elevators
agents	4–4	32–68	29–65	100–136	11–30	6–24	42–60	5–9
goals	16–22	49–85	51–87	120–156	41–59	13–14	62–80	40–40

References

1. Ai-Chang M et al (2004) MAPGEN: mixed-initiative planning and scheduling for the Mars Exploration Rover mission. IEEE Intell Syst 19(1):8–12
2. Bolander T, Andersen MB (2011) Epistemic planning for single and multi-agent systems. J Appl Non-Class Log 21(1):9–33
3. Bonisoli A et al (2014) A privacy-preserving model for the multi-agent propositional planning problem. In: Proceedings of the 2nd ICAPS distributed and multi-agent planning workshop (DMAP-2014), Portsmouth, NH, USA, pp 25–29
4. Borrajo D (2013a) Multi-agent planning by plan reuse. Extended abstract. In: Proceedings of the AAMAS'13, St. Paul, MN (EEUU), pp 1141–1142
5. Borrajo D (2013b) Plan sharing for multi-agent planning. In: *Preprints of the ICAPS'13 DMAP workshop on distributed and multi-agent planning*, Rome, Italy, pp 57–65
6. Borrajo D, Fernández S (2015) MAPR and CMAP. In: Proceedings of the competition of distributed and multi-agent planners (CoDMAP-15), Jerusalem, Israel
7. Borrajo D, Veloso M (2012) Probabilistically reusing plans in deterministic planning. In: Proceedings of ICAPS'12 workshop on heuristics and search for domain-independent planning, Atibaia, Brazil, pp 17–25
8. Bowling M H et al (2003) A formalization of equilibria for multiagent planning. In: Proceedings of IJCAI, pp 1460–1462
9. Brafman R, Zoran U (2014) Distributed heuristic forward search for multi-agent systems. In: Proceedings of the 2nd ICAPS distributed and multi-agent planning workshop (DMAP-2014), Portsmouth, NH, USA, pp 1–6
10. Brafman RI (2015) A privacy preserving algorithm for multi-agent planning and search. In: Proceedings of the twenty-fourth international joint conference on artificial intelligence, IJCAI, pp 1530–1536
11. Brafman RI, Domshlak C (2006) Factored planning: how, when, and when not. In: Proceedings of AAAI-2006, pp 809–814
12. Brafman RI, Domshlak C (2008) From one to many: planning for loosely coupled multi-agent systems. In: Proceedings of ICAPS'08
13. Brafman RI, Domshlak C (2013) On the complexity of planning for agent teams and its implications for single agent planning. Artif Intell 198:52–71
14. Brafman RI et al (2013) Qualitative planning under partial observability in multi-agent domains. In: Proceedings of the 1st workshop on distributed and multi-agent planning (DMAP 2013), pp 26–33
15. Brenner M (2003) A multiagent planning language. In: Preprints of ICAPS'03 workshop on PDDL, pp 33–38
16. Brenner M (2010) Creating dynamic story plots with continual multiagent planning. In: Proceedings of the twenty-fourth AAAI conference on artificial intelligence (AAAI). AAAI Press
17. Brenner M, Nebel B (2009) Continual planning and acting in dynamic multiagent environments. J Auton Agents Multiagent Syst 19(3):297–331
18. Bylander T (1994) The computational complexity of propositional STRIPS planning. Artif Intell 69(1–2):165–204
19. Bäckström C (1998) Computational aspects of reordering plans. J Artif Intell Res 9:99–137
20. Bäckström C, Nebel B (1995) Complexity results for SAS planning. Comput Intell 11(4)
21. Cox JS, Durfee EH (2004) Efficient mechanisms for multiagent plan merging. In: International joint conference on autonomous agents and multiagent systems (AAMAS'04). IEEE Computer Society, Los Alamitos, CA, USA, pp 1342–1343
22. Cox JS, Durfee EH (2005) An efficient algorithm for multiagent plan coordination. In: International joint conference on autonomous agents and multiagent systems (AAMAS'05). IEEE Computer Society, pp 828–835

23. Crosby M et al (2013) Automated agent decomposition for classical planning. In: Borrajo D, Kambhampati S, Oddi A, Fratini S (eds) Proceedings of ICAPS'13. AAAI
24. de Weerdt M, Clement B (2009) Introduction to planning in multiagent systems. Multiagent Grid Syst 5(4):345–355
25. Dimopoulos Y et al (2012) μ-SATPLAN: multi-agent planning as satisfiability. Knowl-Based Syst 29:54–62
26. Durfee EH (1999) Multiagent systems: a modern approach to distributed artificial intelligence, chap. Distributed problem solving and planning. MIT Press, pp 121–164
27. Durkota K, Komenda A (2013) Deterministic multiagent planning techniques: experimental comparison. In: Proceedings of the 1st workshop on distributed and multi-agent planning (DMAP 2013), pp 43–47
28. Ephrati E et al (1995) A tractable heuristic that maximizes global utility through local plan combination'. In: Proceedings of the first international conference on multi-agent systems. MIT Press, San Francisco, USA, pp 94–101
29. Fdez-Olivares J et al (2006) Bringing users and planning technology together. Experiences in SIADEX. In: Proceedings of ICAPS 2006
30. Fernández S et al (2007) PLTOOL. A KE tool for planning and learning. Knowl Eng Rev 22(2):153–184. https://doi.org/10.1017/S0269888907001075
31. Fikes RE et al (1972) Learning and executing generalized robot plans. Artif Intell 3:251–288
32. Foulser D et al (1992) Theory and algorithms for plan merging. Artif Intell 57(2–3):143–181
33. Fox M et al (2006) Plan stability: replanning versus plan repair. In: Proceedings of the sixteenth international conference on automated planning and scheduling (ICAPS'06), pp 212–221
34. García J et al (2013) Combining linear programming and automated planning to solve intermodal transportation problems. Eur J Oper Res 227(1):216–226
35. Gentry C (2010) Computing arbitrary functions of encrypted data. Communications of the ACM, pp 97–105
36. Ghallab M et al (1998) PDDL—the planning domain definition language. Tech. Rep. CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control
37. Ghallab M et al (2004) Automated planning. Theory & Practice, Morgan Kaufmann
38. Guestrin C et al (2003) Efficient solution algorithms for factored MDPs. J Artif Intell Res 19:399–468
39. Helmert M (2006) The fast downward planning system. JAIR 26:191–246
40. Hoffmann J, Nebel B (2001) The FF planning system: fast plan generation through heuristic search. J Artif Intell Res 14:253–302
41. Howey R et al (2004) VAL: automatic plan validation, continuous effects and mixed initiative planning using PDDL. In: Khoshgoftaar TM (ed) ICTAI 2004: 16th IEEE international conference on tools with artificial intelligence, pp 294–301
42. Jonsson A, Rovatsos M (2011) Scaling up multiagent planning: a best-response approach. In: Proceedings of the 21st international conference on automated planning and scheduling (ICAPS'11), pp 114–121
43. Kominis F, Geffner H (2014) Beliefs in multiagent planning: from one agent to many. In: Proceedings of the 2nd ICAPS distributed and multi-agent planning workshop (DMAP-2014), Portsmouth, NH, USA, pp 62–69
44. Kovacs DL (2012) A multi-agent extension of PDDL3.1'. In: Preprints of ICAPS'12 workshop on international planning competition. AAAI Press, Atibaia, Brazil, pp 19–27
45. Langdon AC, Cox MT (2004) The effect of agent topologies on multiagent planning performance. In: Berkowitz EG (ed) Proceedings of fifteenth midwest artificial intelligence and cognitive sciences conference (MAICS). Omni Press, pp 125–129
46. Lesser VR et al (2004) Evolution of the GPGP/TÆMS domain-independent coordination framework. Auton Agent Multi-Agent Syst 9(1–2):87–143
47. Linares López C et al (2013) Automating the evaluation of planning systems. AI Commun 26(4):331–354
48. Lipovetzky N, Geffner H (2012) Width and serialization of classical planning problems. In: ECAI, pp 540–545
49. Long D, Fox M (1999) Efficient implementation of the plan graph in STAN. J Artif Intell Res 10:87–115
50. Luis N, Borrajo D (2014) Plan merging by reuse for multi-agent planning. In: Proceedings of the 2nd ICAPS distributed and multi-agent planning workshop (DMAP-2014), Portsmouth, NH, USA, pp 38–44
51. Maliah S et al (2016) Online macro generation for privacy preserving planning. In: Proceedings of the twenty-sixth international conference on automated planning and scheduling, ICAPS, pp 216–220
52. Milla-Millán G et al (2013) Multi-agent planning based on the dynamic selection and merging of hierarchical task networks. In: Proceedings of the 1st workshop on distributed and multi-agent planning (DMAP 2013, pp 34–42
53. Minton S (1988) Learning effective search control knowledge: an explanation-based approach. Kluwer Academic Publishers, Boston

54. Nissim R, Brafman R (2014) Distributed heuristic forward search for multi-agent planning. JAIR 51:293–332
55. Nissim R, Brafman RI (2013) Cost-optimal planning by self-interested agents. In: Proceedings of AAAI'13
56. Nissim R et al (2010) A general, fully distributed multi-agent planning algorithm. In: Proceedings of 9th international conference on autonomous agents and multiagent systems (AAMAS'10), pp 1323–1330
57. Salido MÁ et al (2009) The container stacking problem: an artificial intelligence planning-based approach. In: Proceeding of the international workshop on harbour, maritime and multimodal logistics modelling and simulation, pp 127–131
58. Scharpff J et al (2013) Planning under uncertainty for coordinating infrastructural maintenance. In: Proceedings of ICAPS'13, pp 425–433
59. Sharon G et al (2012) Conflict-based search for optimal multi-agent path finding. In: Proceedings of AAAI'2012
60. Shehory O, Kraus S (1998) Methods for task allocation via agent coalition formation. Artif Intell 101(1–2):165–200
61. Silver D (2005) Cooperative path finding. In: Young RM, Laird JE (eds) Proceedings of AIIDE, pp 117–122. AAAI Press
62. Smith RG (1980) The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Trans Comput C–29(12):1104–1113
63. Standley T, Korf R (2011) Complete algorithms for cooperative pathfinding problems. In: Proceedings of IJCAI'11, pp 668–673
64. Štolba M et al (2018) Quantifying privacy leakage in multi-agent planning. ACM Trans Internet Technol 18(3):28:1–28:21
65. Szer D et al (2012) MAA*: a heuristic search algorithm for solving decentralized POMDPs. CoRR arXiv:1207.1359
66. Torreño A et al (2014) FMAP: distributed cooperative multi-agent planning. Appl Intell
67. Torreño A et al (2017) Cooperative multi-agent planning: a survey. ACM Comput Surv 50(6):84:1–84:32
68. Torreño A et al (2014) A flexible coupling approach to multi-agent planning under incomplete information. Knowl Inf Syst 38(1):141–178
69. Tozicka J et al (2016) Privacy-concerned multiagent planning. Knowl Inf Syst 48(3):581–618
70. van der Krogt R (2009) Quantifying privacy in multiagent planning. Multiagent Grid Syst 5(4):451–469
71. van der Krogt R, de Weerdt M (2005) Plan repair as an extension of planning. In: Proceedings of ICAPS-05, pp 161–170
72. van Otterloo S (2005) The value of privacy: optimal strategies for privacy minded agents. In: Proceedings of the fourth international conference on autonomous agents and multi-agent systems (AAMAS-05), pp 1015–1022
73. Veloso MM, et al (1990) Nonlinear planning with parallel resource allocation. In: Proceedings of the DARPA workshop on innovative approaches to planning, scheduling, and control. Morgan Kaufmann, San Diego, pp 207–212
74. Vickrey W (1961) Computer speculation, auctions, and competitive sealed tenders. J Finance 16:8–37
75. Virseda J et al (2014) Multi-agent planning with agent preferences. In: Proceedings of the 2nd ICAPS distributed and multi-agent planning workshop (DMAP-2014), Portsmouth, NH, USA, pp 70–78
76. Wallace RJ, Freuder EC (2005) Constraint-based reasoning and privacy/efficiency tradeoffs in multi-agent problem solving. Artif Intell 161(1–2):209–227
77. Walsh WE, Wellman MP (2003) Decentralized supply chain formation: a market protocol and competitive equilibrium analysis. J Artif Intell Res 19:513–567
78. Wang X (1994) Learning planning operators by observation and practice. In: Proceedings of the second international conference on AI planning systems, AIPS-94, AAAI Press, Chicago, IL, pp 335–340
79. Yang Q et al (2007) Learning action models from plan examples using weighted MAX-SAT. Artif Intell 171(2–3):107–143

Daniel Borrajo received his Ph.D. from Universidad Politécnica de Madrid (1990). Currently, he is a Professor of Computer Science at Universidad Carlos III de Madrid. He has published over 200 journal and conference papers, mainly in the fields of problem solving methods (heuristic search and automated planning) and machine learning. He has been the Program Co-chair of the International Conference of Automated Planning and Scheduling (ICAPS'13), Conference co-chair of the Symposium of Combinatorial Search (SoCS'12, SoCS'11) and ICAPS'06, and PC member (including Area chair and Senior PC) of the main conferences on Artificial Intelligence (e.g., IJCAI, AAAI, ICAPS, ...). His current research interests lie in goal reasoning, multi-agent planning, and machine learning applied to planning.

Susana Fernández is currently an Associate Professor at the Universidad Carlos III de Madrid, since 2007. She received the B.A. degree in Physics (Universidad Complutense de Madrid, 1989), and a M.Phil. degree in "Logic, Text and Information Technology" (University of Dundee, Scotland, 1991). She worked for 10 years as a software engineer in two private companies, INDRA and Eliop, S.A. She received the Ph.D. degree in Computer Science (Universidad Carlos III de Madrid, 2006). Her current research interest is in Artificial Intelligence, particularly automated planning, machine learning and multi-agent systems. She has participated in several competitive projects related to automated planning, problem solving or agents. She has published more than 20 papers.