Multi-Agent Planning with Agent Preferences

Jesús Virseda and **Susana Fernández** and **Daniel Borrajo**
Departamento de Informática
Universidad Carlos III de Madrid
jvirseda@inf.uc3m.es, sfarregu@inf.uc3m.es, dborrajo@ia.uc3m.es

Abstract

In the field of Automated Planning, there is a renewed interest in Multi-Agent Planning (MAP). In this paper, we focus on the task of planning efficiently when agents have preferences over goals and want to maintain privacy. Our approach takes as input a MAP task, and a preference of each agent for each goal. Those preferences can be mapped into utilities (rewards) and are modeled as costs (penalties) when planning. The planner has to generate a valid plan taking into account the agent's preferences and the cost of actions. We compare different approaches for assigning goals to agents that heed either the expected costs or preferences, and plan considering either costs or preferences.

Introduction

In the field of Multi-Agent systems, there has been plenty of work on self-interested agents. Most works focus on negotiation, auctions, or single-decision tasks. In Multi-Agent Planning (MAP), planners should also take into account the fact that they have to generate a valid plan (where a sequence of actions is involved, which is different to other sequences of decisions in Multi-Agent systems, as repeated auctions). In MAP, there has not been many approaches yet that focus on self-interested agents. Current work by Nissim and Brafman (2013) focused first on optimal planning, minimizing the cost of plans, to then compute *a posteriori* the payoff (utility) of agents participating in a plan. This payoff is, in other words, a fair payment for each agent proportionally according to its contribution based on plan costs. So, it does not take into account preferences that agents might have *a priori* on achieving specific goals.

In this paper, we focus on a different task, which is based on several past and current projects of our group, where each agent has private information and preferences over goals. We will call it the Multi-Agent Planning with Preferences (MAPP) task. For instance, in logistics transportation domains, each branch of a transportation company has preferences over goals (moving some goods at some place) since they might prefer a transport task over another (García et al. 2013). Likewise, in an ESA (European Space Agency) project, sensors must be assigned (as satellites or antennas) to observe objects based also on some preferences of sensors over objects (Arregui et al. 2012). Finally, in two more projects we dealt with tourist plans where each user has a set of preferences over goals (places to visit) (Cenamor, de la Rosa, and Borrajo 2013; Castillo et al. 2008). So, our main aim is to efficiently find good-quality plans that maximize preference over goals while maintaining agents privacy.

Here, we propose an approach for MAP where agents have explicit preferences over goals, which can be independent of costs to achieve the goals. Solutions ideally would minimize the cost while maximizing the compliance with the preferences of all agents. Here, we extend the distributed approach proposed in (Borrajo 2013b; 2013a) (MAPR) for solving MAP tasks. MAPR first assigns goals to agents and then iteratively solves each agent problem preserving the privacy of agents during the process. Moreover, MAPR flexibility allows us to use any state-of-the-art planner.

In order to reason with preferences, we have to model them into planning tasks. Preferences can be implemented in a positive or negative way. That is to say, if preferences over goals are seen as *rewards* for the agents which achieve the goals, they are positive preferences; and if preferences are seen as *penalties*, they are negative preferences. Since most planners work minimizing total-cost metrics, we use the negative implementation of preferences.

Thus, the problem of MAP with preferences can be understood as a multi-criteria optimization problem, where two different and not related metrics must be minimized: cost and penalty. In our approach, we work with two PDDL domain files and their respective problem files. The *cost* domain and problem model the original planning task, ignoring agent preferences. And, the *penalty* domain and problem ignore action costs. In this latter case, penalties are added in the effects of the actions that achieve goals. We study here the impact in terms of runtime, cost and reward of using both metrics at two different steps in the algorithm: when assigning goals to agents; and during search while the planner is solving each agent MAP task.

The next sections formalize the MAPP task, describe the employed MAPR approach for distributed MAP, give the details of the preferences over goals compilation as penalties, explain the experimental setup and show the results, contrast the differences with related work and finalize with the conclusions of this work.

Multi-Agent Planning Tasks with Preferences

We are interested in MAP tasks with Preferences (MAPP) that can be formally defined as:

Definition 1. *A MAPP task for a set of self-interested agents* $\Phi = \{\phi_i\}_{i=1}^m$ *with private and public information is a tuple* $\Pi = \{F, A, I, G, c, p\}$ *where:*

- $F = F_P \cup \{F_i\}_{i=1}^m$ is a set of fluents that can be public, F_P, or private for each agent, $\{F_i\}_{i=1}^m$
- $A = \{A_i\}_{i=1}^m \cup A_E$ is the set of instantiated actions that agents can perform, $\{A_i\}_{i=1}^m$, and the optional set of *external* actions, A_E, performed by other agents not in Φ
- $I \subseteq F$ is an initial state
- $G \subseteq F$ is a set of goals
- $c : A \rightarrow \Re$ is a cost function representing the cost of every action $a \in A$
- $p : \Phi, G \rightarrow \Re$ is a preference function representing the preference that agent $\phi \in \Phi$ has for achieving goal $g \in G$

Each action $a \in A$ is described by a set of parameters (par(a)), 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 effects) or removed (delete effects) from the state after the execution of the action. The actions of an agent ϕ_i can be described as: $A_i = \{a \in A \mid \phi_i \in par(a)\}$. And external actions are: $A_E = \{a \in A \mid par(a) \cap \Phi = \emptyset\}$.

The planning task should generate as output a sequence of actions $\pi = (a_1, \ldots, a_k)$ such that if applied in order would result in a state s_k where goals are true: $G \subseteq s_k$. We will later define the plan cost.

A MAPP task Π can be naturally decomposed into a set of partial planning subtasks, one for each agent $\{\Pi_i^p\}_{i=1}^m$ as:

- $\Pi_i^p = \{F_i^p, A_i^p, I_i^p, G_i^p\}$
- $F_i^p = F_P \cup F_i$
- $A_i^p = A_i \cup A_E$
- $I_i^p = I \cap (F_i \cup F_P)$
- $G_i^p = G \cap (F_i \cup F_{P_i})$

where F_{P_i} is the subset of public goals assigned to agent ϕ_i: $\forall i, F_{P_i} \subseteq F_P$ and $\cup_{i=1}^m F_{P_i} = G \cap F_P$. We will see later how we assign a subset of the public goals to each agent. As explained in the next section, our MAP algorithm solves Π by iteratively solving a subset of planning subtasks $M = \{\Pi_1^{p+}, \ldots, \Pi_n^{p+}\}$, $n \leq m$ (since not all agents will need to plan). Each Π_i^{p+} completes Π_i^p with the information communicated by the previous agents in the iteration.

In our work, we consider that privacy is directly related to the information on the state and goals that agents have on a particular domain and problem. Others consider that actions are also public or private, which is not our case. It is only the information on preconditions and effects of actions that is private or public (Nissim and Brafman 2013). Preserving privacy in our work means that agents solve their planning subtasks without ever knowing the private information of other agents. Thus literals $l \in F$ are considered either private or public. In case they are private, they belong to a given agent ϕ_i and they should only be known and modified by ϕ_i when planning. In particular, literals $l \in I$ and $l \in G$ can be private or public in turn. In order to maintain privacy, our planner obfuscates some planning components when agents finish their planning episodes, and communicate them to the following agents, as explained later.

As an example, in the Transport domain of the International Planning Competition (IPC)[1] several vehicles (agents) must transport packages among locations. Fluents derived from the PDDL predicates *(at ?x - vehicle ?v - location)* and *(in ?x - package ?v - vehicle)* and from the function *(capacity ?v -vehicle)* are private. The cost of the action *drive* depends on the road length and the other actions have a cost of 1. In this IPC domain, problems goals only derive from the (at ?x - package ?v - location) predicate. Since the action *drop* is the only one that achieves the *at* predicate, *drop* is the only action achieving goals of Transport problems, and thus the only one that provides rewards to agents. The function p defines the preference every vehicle has for *dropping* every package involved in the goals. There are no external actions in this domain. In other domains, as in the Driverlog where the agents are the drivers, there are external actions, such as *load-truck* and *unload-truck*, since no agent (driver) intervenes in any of the two.

MAPR

MAPR automatically generates the partial planning subtasks $\{\Pi_i^p\}_{i=1}^m$ from the PDDL description of a domain and problem and from the agents description (public goals are assigned to agents at this point). Next, the MAPR algorithm iteratively solves each agent problem. [2] Once an agent solves a problem, it obfuscates the private components of the solution and communicates them to the next agent. 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. Therefore, MAPR sees MAP as plan reuse. An important aspect of the algorithm consists on how to assign public goals to agents. As we will explain below, it uses several standard strategies.

Figure 1 shows a high-level description of the algorithm, where we use @ to express obfuscated private information. It takes as input a MAP task (domain, problem and agents description), a goal assignment strategy, 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 (that could be different) is that the second planner might be a replanning system. Since all inputs and outputs are in PDDL, MAPR can use any state-of-the-art planner. The algorithm is then composed of six main steps: goal assignment; first planning episode; obfuscation of the private part of a plan and communicating information to the next agent; merging of a prior agents plan with a planning problem; subsequent planning episodes; and termination. The goal assignment strategy may not assign goals to some of

[1] http://ipc.icaps-conference.org/

[2] For a more detailed description, we refer the reader to (Borrajo 2013b).

the agents and thus these agents are not used in the planning process. As a side comment on the algorithm, in the second and following iterations, when $j = 1$, then $j-1$ means $j = n$ ($n \leq m$). So, the first agent, instead of generating a new plan using the first planner, it takes as input the obfuscated solution from the last agent on the previous iteration. Since MAPR can iterate over each agent once it has completed the first iteration over all agents, MAPR benefits from an implicit backtracking. So, if in the first iteration an agent could not complete its goals due to a wrong decision (such as using a particular action for achieving a goal, or consuming a common resource that another agent needs), MAPR could potentially find a solution if it can be generated by the set of chosen agents.

Function MAPR (M, GA, FP, SP): plan

M: multi-agent planning task
GA: goal assignment strategy
FP: first planner
SP: second planner

Assign subset of public goals to each agent ϕ_i using GA
$\pi_1 \leftarrow$**First-Plan**(FP, Π_1^p)
$j \leftarrow 1$
Repeat until **Termination**
  $j \leftarrow j + 1$
  If $j > n$ Then $j \leftarrow 1$
  ϕ_{j-1} **Obfuscates** its private information, $S_{j-1}^{@}$:
    • the plan $\pi_{j-1}^{@}$ and
    • the problem $\Pi_{j-1}^{@} = \{F_{j-1}^{@}, A_{j-1}^{@}, I_{j-1}^{@}, G_{j-1}^{@}\}$
  ϕ_{j-1} **Communicates** $S_{j-1}^{@}$ to agent ϕ_j
  ϕ_j creates a new planning task, Π_j^{p+}:
    • it **Merges** its assigned problem and $S_{j-1}^{@}$
  $\pi_j \leftarrow$**Second-plan**(SP, Π_j^{p+})
If solved, return last plan

Figure 1: High level description of MAPR planning algorithm.

Goal Assignment

Given the total set of public goals G and a set of agents Φ, MAPR first has to assign a subset of goals to each agent to lower the planning complexity of each individual planning episode. For each goal in $g \in G$ and agent in $\phi_i \in \Phi$, MAPR computes a relaxed plan from the initial state of each agent, I_i, following the well known relaxed plan heuristic of FF (Hoffmann and Nebel 2001). If the relaxed plan heuristic detects a dead-end, then $c(g, \phi_i) = \infty$. This will define a cost matrix, $c(G, \Phi)$. Next, we have devised four goals assignment schemes.

all-achievable: MAPR 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 ϕ_i, g is assigned to ϕ_i.

rest-achievable: MAPR 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.

best-cost: MAPR assigns each goal g to the agent that can potentially achieve it with the least cost, $\arg\min_{\phi_i \in \Phi} c(g, \phi_i)$

load-balance: MAPR tries to keep a good work balance among agents. It first computes the average number of goals per agent, $k = \frac{|G|}{m}$. 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, $\mid G \mid -k \times (m-1)$.

In configurations rest-achievable and best-cost, there can be agents for which MAPR does not assign goals.

Obfuscation

If an agent ϕ_j solves its subproblem, then it cannot pass the private information openly to the next agent. So, it obfuscates[3] the private parts and communicates an augmented obfuscated solution $S_j^{@}$ to the next agent. There can be potentially many algorithms for obfuscating the information. In this paper, we use the same simple version of this procedure described in (Borrajo 2013b). Depending on the privacy commitment of the planning task, more complex obfuscating algorithms could be used and the difference will be: more time devoted to the obfuscating algorithm (their time complexity is usually much less than the one of planning); and potentially more space of the obfuscated information (any obfuscating algorithm with a space polynomial complexity could be used without affecting the overall multi-agent planning complexity).

In our case, obfuscating is a two steps process. First, a random substitution is generated for the names of all private predicates, actions and objects. Action names are obfuscated given that, in our privacy preserving scheme, other agents do not need to know the specific actions used by any agent to achieve the goals, even if all information used by those actions is public. As a reminder, in our privacy preserving scheme, actions are not considered public or private; it is only the propositions that are private or public. For instance, in the Satellite domain, if a plan contains an instantiated action as `(calibrate sat1 inst1 Phen6)`, given that `calibrate` and `inst1` are private, MAPR would generate a random substitution as: [4]

$\sigma =$ {`(calibrate . g12) (inst1 . g23)`},
resulting in `(g12 sat1 g23 Phen6)`

The second step in obfuscation consists of applying the substitution to the plan. An augmented obfuscated solution $S_j^{@}$ consists of the obtained plan and the set of components that are needed by the rest of agents to regenerate that solution. More specifically, if the plan of ϕ_j is $\pi_j = (a_1, \ldots, a_t)$, it communicates $S_j^{@} = \{\pi_j^{@}, A_j^{@}, I_j^{@}, G_j^{@}\}$ to ϕ_{j+1}:

[3] We will use obfuscate indistinctly of encrypt.

[4] We are describing the process in the PDDL lifted version, instead in the propositional version.

- the set of instantiated actions in the plan, after obfuscating them, $A_j^{@}$, by obfuscating the actions parameters (par(a_i)), preconditions (pre(a_i)), and effects (eff(a_i)):
$A_j^{@} = \{a_i^{@} \mid a_i \in \pi_j, a_i^{@} = (\text{par}(a_i)\mid_\sigma, \text{pre}(a_i)\mid_\sigma, \text{eff}(a_i)\mid_\sigma)\}$
where we use the notation $\alpha\mid_\sigma$ to represent the result of applying substitution σ to formula α.
- the obfuscated plan, $\pi_j^{@} = \{a_1^{@}, \ldots, a_t^{@}\}$, since we can use planning by reuse in the next iteration instead of planning from scratch.
- all goals (private and public, including goals of previous agents), after obfuscating the private ones,[5]

$$G_j^{@} = \{g^{@} \mid g \in G_j, g^{@} = g\mid_\sigma\}$$

- initial state, after obfuscating the private information. 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:

$$I_j^{@} = \{f^{@} \mid f \in I_j, a_i \in \pi_j, f \in \text{pre}(a_i), f^{@} = f\mid_\sigma\}$$

Planning with Preferences

In MAPP tasks, there are two independent metrics: cost and preferences. We deal with cost as in regular planning settings.

Definition 2. *The cost of a plan π is defined as: $C(\pi) = \sum_{a_i \in \pi} c(a_i)$.*

Preferences cannot be handled directly by current planners, as they can only minimize metric values (in fact, most current planners only allow the definition of one metric in a domain file, named `total-cost`). Thus, we have to map preferences to a minimizing criteria, as penalty. Preferences are defined for goals and agents, while metrics are defined in actions. Thus, we have to map preferences into action costs. In order to define the mapping, we first translate agent-goals preferences into actions rewards.

Definition 3. *The reward $r(a_i, g_k, \pi)$ an action a_i receives for achieving a goal g_k in plan π is defined as:*

$$r(a_i, g_k, \pi) = \begin{cases} p(\phi_j, g_k) & \text{if } \phi_j \in \textit{par}(a_i) \text{ and } a_i \text{ is the last achiever of goal } g_k \text{ in } \pi \\ 0 & \text{otherwise} \end{cases}$$

So, actions only receive a reward if they are the only ones that achieve a top-level goal and there is an agent that executes it. We are assuming that the preference relation is defined for each agent and goal. Now, we can define the total reward that an action receives.

Definition 4. *The total reward an action a_i receives in a plan π is defined as:*

$$r(a_i, \pi) = \sum_{g_k \in G, g_k \in \textit{eff}(a_i)} r(a_i, g_k, \pi)$$

Now, the reward of a plan is the sum of all rewards obtained by the preferences for goals of the agents that achieved those goals.

[5]Substitution only affects the private goals.

Definition 5. *The total reward $R(\pi)$ of a plan π is defined as $R(\pi) = \sum_{a_i \in \pi} r(a_i, \pi)$.*

Since most planners minimize total-cost metrics, in this paper preferences are converted into penalties (negative preferences) in a standard way.

Definition 6. *The penalty $\rho(a_i, g_k, \pi)$ an action a_i receives for achieving a goal g_k in plan π is defined as:*

$$\rho(a_i, g_k, \pi) = \begin{cases} r_{\max} - r(a_i, g_k, \pi) & \text{if } a_i \text{ is the last achiever of goal } g_k \text{ in } \pi \\ 0 & \text{otherwise} \end{cases}$$

where $r_{\max}$ is the maximum possible reward.

Definition 7. *The total penalty an action a_i receives in a plan π is defined as:*

$$\rho(a_i, \pi) = \sum_{g_k \in G, g_k \in \textit{eff}(a_i)} \rho(a_i, g_k, \pi)$$

We are interested in agent preferences for achieving goals, so we implement the penalties as increments of the total-cost function only in the actions that achieve goals when they achieve a goal predicate. For example, in the Transport domain where goals derive from the *at* predicate and only the *drop* action has *at* as a positive effect, a new effect *(increase (total-cost) (penalty-vehicle-at ?v ?l ?p))* is added. The init part of the problem contains the values of the *penalty-vehicle-at* function. Only instantiations with the same parameters as some of the goals have values different from 0.

Now, we can define the total penalty of a plan (equivalent to a reward obtained by fulfilling the agents preferences).

Definition 8. *The total penalty $P(\pi)$ of a plan π is defined as $P(\pi) = \sum_{a_i \in \pi} \rho(a_i, \pi)$.*

Note that $r(a_i, g_k, \pi)$ (and consequently $\rho(a_i, g_k, \pi)$ too) includes the condition that a_i is the last one in the plan π that achieves a goal g_k. Thus, this definition is plan dependent. In this paper, we are interested in domains where goals need to be achieved only once and it is not necessary to undo achieved goals. Hence, we can assume that $r(a_i, g_k, \pi) \simeq r(a_i, g_k)$ and $\rho(a_i, g_k, \pi) \simeq \rho(a_i, g_k)$.

Since most state-of-the-art planners only allow the minimization of the *total-cost* function as the unique metric (as, for instance, all planners based on FAST-DOWNWARD (Helmert 2004)), we define two domains for each planning task. As a side note, this is interesting given that the initial idea of defining metrics in PDDL was that domains could reason on different metrics (so all problems in a given domain would use the same domain file) and it would be in the problem where one would define which metric to use for that particular problem. Now, we are forced to define N different domains, one for each metric, while we only need one problem! The first domain is the original one, ignoring agents preferences. And the second domain implements penalties as a total-cost metric. The problem task is common for both domains: the original problem enriched with the penalty information.

Formally, we can say that given a MAPP task $\Pi = \{F, A, I, G, c, p\}$, a MAPP task Π' with *penalty* costs can be obtained by the following transformation:

Definition 9. *Given a MAPP task with action costs and preferences over goals* $\Pi = \{F, A, I, G, c, p\}$ *the equivalent MAPP task with penalty costs can be defined as* $\Pi' = \{F', A, I', G, c'\}$ *with:*

- $F' = F \cup F_p$*, where* $F_p = \{\rho_{ij} | g_i \in G, \phi_j \in \Phi\}$*. Each* ρ_{ij} *is a PDDL function representing the penalty agent* ϕ_j *has for achieving goal* g_i*.*
- $I' = I \cup F_p$
- $G' = G \cup F_p$
- $c' : A \rightarrow \Re^+$ *is a new cost function defined as:*

$$c'(a) = \rho(a, g_i)$$

Therefore, given a MAPP task $\Pi = \{F, A, I, G, c, p\}$, we work with two MAPP tasks: Π' with penalty costs as defined in Definition 9 and $\Pi'' = \{F, A, I, G, c\}$ with standard costs and no preferences. Then, we use the extended MAPR algorithm following four strategies: use Π'' for assigning goals to agents and for planning (cc); use Π' for both (pp); use Π' for assigning goals and Π'' for planning (pc); and use Π'' for assigning goals and Π' for planning (cp). And we can use different metrics to evaluate the quality of the generated plans: plan cost $C(\pi)$, plan reward $R(\pi)$, plan penalty $\rho(\pi)$ and plan utility $U(\pi) = R(\pi) - C(\pi)$ that relates the reward and the cost. In unit-cost domains, plan cost is the length of the solution plan. All measures apply to all strategies. Even if strategies cp and pp ignore action costs when planning, it is possible to compute $C(\pi)$ *a posteriori* by consulting the costs $c(a_i)$ in π. Penalty and reward are opposed values, so it is possible to compute one value based on the other. Also, even if goals are assigned by the agents preferences/costs (so it selects agent ϕ_j because it is the one with highest-preference/lowest-cost over goal g_i) MAPR does not force that in the final plan it is in fact agent ϕ_j that achieves g_i.

This approach is valid only when the goals need to be achieved once. Thus, the actions that are late achievers of goals will be the only achievers of goals (definition of ρ depends on the plan). However, in domains like the Sokoban an agent could place a stone in a goal position, and then another agent might have to move the stone to a different position to fulfill all problem goals. In this case, the reward obtained by the first agent when it places the stone in the temporal goal position should be subtracted from the total reward, so only late achievers get credit. An alternative way to solve a MAPP task Π that avoids this problem is to transform Π into a new task with *soft goals* and *negative utilities* and then compile them away using the technique described in (Keyder and Geffner 2009). Negative utilities stand for conditions to be avoided; for example, a utility $u(p \wedge q)$ = -10 penalizes a plan that results in a state where both p and q are true with an extra cost of 10. We are not using this definition in this paper, since the domains we used in the experiments do not have this problem. However, we provide at least the solution to this problem here. The MAPP task Π_S with soft goals and negative utilities that is equivalent to Π can be obtained by the following transformation:

Definition 10. *Given a MAPP task* $\Pi = \{F, A, I, G, c, p\}$*, the equivalent MAPP task with soft goals and negative utilities is defined as* $\Pi_S = \{F_S, A_S, I, G_S, c, \nu\}$ *where:*

- $G_S = G \cup \{G_p\}_{i=1}^n$*, where* $\{G_p\}_i = \{\gamma_{ij} | g_i \in G, \phi_j \in \Phi\}$ *are the new soft goals* γ_{ij} *representing that agent* ϕ_j *achieves the goal* g_i *with its corresponding preference value* $p(\phi_j, g_i)$
- $F_S = F \cup \{G_p\}_{i=1}^n$
- A_S *transforms every action* $a \in A | (g_i \in \mathit{eff}(a)) \wedge (g_i \in G) \wedge (\phi_j \in \mathit{par}(a))$ *by adding as a new effect the soft goal* $\gamma_{ij} \in \{G_p\}_i$
- $\nu : \{G_p\}_{i=1}^n \times \{G_p\}_{i=1}^n \rightarrow R^-$ *is the negative utility function defined over every pair* $(\gamma_{ij}, \gamma_{ik})$ *in the following way* $\nu(\gamma_{ij} \wedge \gamma_{ik}) = -p(\phi_j, g_i)$

Properties

The approach we propose to solve preference problems in MAPP inherits the properties of MAPR, i.e. it is suboptimal, sound and incomplete.

Experiments and Results

We have used the following experimental setup for comparison:

Comparing approaches. We compare different distributed strategies against a centralized approach. These distributed strategies are the ones previously defined (cc, pc, cp, and pp). We only show a centralized approach to understand the relation between a distributed approach which preserves privacy and a centralized one that does not preserve privacy. Thus, the centralized approach has an advantage over the distributed approach. The centralized approach is LAMA11, the winner of the last IPC (Helmert 2004; Richter and Westphal 2010). We are interested here on efficient MAPP computation. Therefore, we have used only the first search iteration of LAMA11, that is one run of lazy greedy best first search with actions costs, and FF and LM-cut heuristics with preferred operators. We plan to move into optimal planning or at least improve the quality of the solutions with anytime behavior in the future.

Domains. We have chosen four domains from the previous IPCs that have been regularly used in MAP papers: Rover, Satellite, Transport and Zenotravel. This selection has been done according to our main motivation: domains close to real world problems where agent preferences are relevant. The Transport domain implements action-costs while the other three are unit-cost. The maximum penalty / reward are set as 10 in all of them, because it seems to be intuitive to ask users for preferences in scales from zero to ten. Costs are the original ones used in the IPC and penalties have been generated randomly to ensure independence from the cost values.

Goal assignment. We have used the four defined methods: all-achievable, rest-achievable, load-balance and best-cost.

Planners. We have used the centralized approach explained above for generating the first agent plan and for the successive planning episodes too.

Time and memory bounds. We have used 1800 seconds and 6GB RAM as in the IPC.

Scores. We have used the following metrics, similar to those used in the IPC, to compare the different approaches:

- *Coverage* is the number of solved problems by each approach.
- *Runtime score (S_T)* over a set of problems $\mathcal{P}$, assuming that $T_p^* > 0$, is computed as

$$S_T = \sum_{p \in \mathcal{P}} \frac{1}{1 + \log \frac{T_p}{T_p^*}}$$

  where T_p^* is the minimum time required to solve the problem p by any approach, and T_p is the time required by the approach we want to calculate the score.
- *Cost score (S_C)* over a set of problems $\mathcal{P}$ is computed as

$$S_C = \sum_{p \in \mathcal{P}} \frac{C_p^*}{C_p}$$

  where C_p^* is the minimum cost of any solution of the problem p and C_p is the cost of the solution by the approach we want to calculate the score. This scores is called quality score in the IPC.
- *Reward score (S_R)* over a set of problems $\mathcal{P}$ is calculated as

$$S_R = \sum_{p \in \mathcal{P}} \frac{R_p}{R_p^*}$$

  where R_p^* is the maximum reward achieved in any solution of the problem p and R_p is the reward obtained by the approach we want to calculate the score.
- *(Reward, Cost) pareto-dominance* gives to each approach a score that equals the number of tuples it pareto-dominates for the same problem. (R, C) is said to pareto-dominate (R', C') if and only if $R \geq R'$ and $C \leq C'$. We use this score instead of a utility score to avoid having to normalize the reward and cost values.

We prefer (Reward, Cost) pareto-dominance instead of a utility score (reward - cost), because the utility subtracts two amounts in different metrics. Therefore, the weight of both metrics on the score depends on the domain / problem; whether the plan solutions are long or short, or whether the actions costs are high or not in comparison to the number of goals to achieve, which sets the maximum reward achievable in the problem.

Tables 1, 2, 3 and 4 show the runtime, cost and penalty score results for the domains Rover, Satellite, Transport and Zenotravel, respectively. Table 5 summarizes all the score results in a table.

In terms of coverage, the only domain that presents difficulties is the Transport domain, where only the configurations of the rest-achievable goal selection can solve all problems. However, configurations that use costs when planning with the best-cost goal selection obtain good results in coverage too.

The fastest approach in all domains but Rover uses penalties and the rest-achievable strategy during goal selection and costs for planning. Only the centralized approach using cost metric outperforms it in the Rover domain. Globally, the rest-achievable strategy is the best one when we want to find a solution fast.

Table 1: Results in the Rover domain.

coverage	pc	pp	cp	cc
best-cost	20	20	20	20
load-balance	20	20	20	20
rest-achievable	20	20	20	20
all-achievable	20	20	20	20
centralized		20		20
S_T	pc	pp	cp	cc
best-cost	16.06	15.36	16.74	16.14
load-balance	15.16	14.66	15.01	14.73
rest-achievable	**18.36**	17.31	18.14	17.44
all-achievable	14.93	14.44	14.79	14.53
centralized		18.60		**18.96**
S_C	pc	pp	cp	cc
best-cost	18.00	17.62	18.19	19.18
load-balance	18.73	17.90	17.90	18.73
rest-achievable	18.08	17.29	17.29	18.08
all-achievable	**19.35**	17.83	17.83	**19.35**
centralized		18.41		**19.64**
S_R	pc	pp	cp	cc
best-cost	16.53	19.57	16.62	14.11
load-balance	14.13	17.94	17.94	14.13
rest-achievable	14.64	17.03	17.03	14.64
all-achievable	14.23	**19.77**	**19.77**	14.23
centralized		19.63		14.45
(R, C) pareto-d.	pc	pp	cp	cc
best-cost	85	96	101	76
load-balance	65	95	93	60
rest-achievable	59	69	64	54
all-achievable	62	104	**108**	57
centralized		**149**		87

The best quality plans in all domains, except for the Transport, are obtained by the centralized approach using the cost metric. In the Transport domain, surprisingly, the best configurations are the ones that select goals with the rest-achievable strategy and plan with penalties. The score in the Transport domain by the rest-achievable strategy is clearly influenced by its high coverage. On the other hand, the Transport domain is the only one that implements action-costs (the other three are unit-cost) and the first solution is not the most relevant one to compare the quality of the approaches. Furthermore, the rest-achievable goal selection strategy distributes goals without taking into account the costs / penalties, pruning the search. In the other strategies, the best configurations are the ones that plan using costs, as it was expected. The best-cost goal selection strategy obtains good results in terms of the cost score when it selects the goals using the action-costs and plans with the same metric.

The reward score is the most diverse of all. Globally, the best approach for this metric is the centralized approach using penalties in planning tasks. In the Rover domain, though, the best configurations use the all-achievable goal selection, and plan with penalties. In the Transport domain the best configuration uses the rest-achievable goal selection strategy, and in the Zenotravel domain the best-cost goal selection

Table 2: Results in the Satellite domain.

coverage	pc	pp	cp	cc
best-cost	20	20	20	20
load-balance	20	20	20	20
rest-achievable	20	20	20	20
all-achievable	20	20	20	20
centralized		20		20
S_T	pc	pp	cp	cc
best-cost	15.96	14.16	15.78	16.61
load-balance	15.62	13.87	14.23	15.20
rest-achievable	**19.16**	16.54	17.28	18.24
all-achievable	15.27	13.64	13.89	14.89
centralized		13.81		18.14
S_C	pc	pp	cp	cc
best-cost	17.87	16.67	17.10	17.85
load-balance	17.64	16.72	16.72	17.64
rest-achievable	17.81	16.06	16.06	17.81
all-achievable	**18.01**	15.73	15.73	**18.01**
centralized		16.28		**19.29**
S_R	pc	pp	cp	cc
best-cost	10.76	**17.43**	14.61	9.23
load-balance	10.76	14.99	14.99	10.76
rest-achievable	6.19	12.38	12.38	6.19
all-achievable	9.26	16.55	16.55	9.26
centralized		**17.98**		10.01
(R, C) pareto-d.	pc	pp	cp	cc
best-cost	92	**102**	86	75
load-balance	77	80	81	79
rest-achievable	48	49	45	46
all-achievable	61	57	56	60
centralized		106		**118**

Table 3: Results in the Transport domain.

coverage	pc	pp	cp	cc
best-cost	18	15	17	19
load-balance	16	14	14	17
rest-achievable	**20**	**20**	**20**	**20**
all-achievable	13	12	11	13
centralized		14		17
S_T	pc	pp	cp	cc
best-cost	10.56	7.51	9.79	12.99
load-balance	10.25	7.16	7.19	10.51
rest-achievable	**19.66**	15.90	16.04	18.52
all-achievable	6.35	5.38	4.96	6.37
centralized		5.64		8.42
S_C	pc	pp	cp	cc
best-cost	13.19	10.38	13.11	15.42
load-balance	12.34	10.06	10.06	13.00
rest-achievable	16.16	**17.30**	**17.30**	16.16
all-achievable	9.76	7.52	6.76	9.76
centralized		9.63		13.09
S_R	pc	pp	cp	cc
best-cost	12.38	11.29	12.22	12.92
load-balance	9.74	9.47	9.47	10.47
rest-achievable	**15.04**	15.03	15.03	**15.04**
all-achievable	7.37	8.71	7.79	7.37
centralized		13.98		12.73
(R, C) pareto-d.	pc	pp	cp	cc
best-cost	59	43	75	68
load-balance	30	39	37	42
rest-achievable	122	**128**	**128**	123
all-achievable	12	16	12	15
centralized		61		83

using penalties to select the goals and to plan.

Finally, the (Reward, Cost) pareto-dominance indicates which are the best balanced approaches; those which dominate more often the other ones in both metrics: reward and cost. The best balanced approach is the centralized one, specifically when it uses the costs. In the Transport domain the rest-achievable approaches obtain the best results influenced by the coverage. In the Rover domain, approaches which use penalties to plan are significantly better than those which use costs to plan. On the opposite side is the Zenotravel domain, where the best approaches are which uses costs to plan. In the rest of domains, the results are similar.

Table 5 shows that most MAPP configurations score higher than the centralized approach in runtime. In the other metrics, at least one of the MAPP configurations obtained a score close to the centralized approach. As a reminder, the centralized approach cannot be directly compared against our configurations, given that it does not preserve privacy.

Analyzing the results in more depth, we can affirm that the distributed approaches become to outperforms the centralized ones when the problems come to be more difficult. This fact can be observed in the Transport domain, the most difficult one, where distributed approaches obtain the best results in all the scores. Additionally, approaches that plan using cost get better results in terms of cost score and approaches that plan using penalties get better reward score results. If the goal selection step employs either cost or penalty metric does not seem to affect so much these scores.

Related Work

Most work on multi-agent planning for self-interested agents focuses on finding *stable* solutions in the spirit of game theory (Brafman et al. 2009; Crosby and Rovatsos 2011). Agents have their own goals and are able to form coalitions, costless binding agreements, to fulfill them. A stable solution is a coalition's joint plan such that no subset of its agents would benefit by joining an alternative coalition. Agents' payoffs are computed *a posteriori* and depend on the total cost of the actions carried out by the agents. Nissim and Brafman proposed a privacy-preserving distributed mechanism to find cost optimal solutions that also calculates the payments to each agent (Nissim and Brafman 2013). We model the *self interest* of the agents with the preference function, independently of the cost. And, we calculate suboptimal and not stable solutions.

Oversubscription planning problems also define preferences on the goals (Keyder and Geffner 2009; Smith 2004). They assume it is not possible to achieve all *soft* goals due to

Table 4: Results in the Zenotravel domain.

coverage	pc	pp	cp	cc
best-cost	20	20	20	20
load-balance	20	20	20	20
rest-achievable	20	20	20	20
all-achievable	20	20	20	20
centralized		20		20
S_T	pc	pp	cp	cc
best-cost	15.06	13.73	15.91	16.48
load-balance	14.96	13.80	14.06	14.67
rest-achievable	**19.93**	17.97	18.61	19.11
all-achievable	14.74	13.57	13.76	14.24
centralized		14.19		15.77
S_C	pc	pp	cp	cc
best-cost	16.03	14.60	15.95	17.73
load-balance	16.00	14.54	14.54	16.00
rest-achievable	17.35	15.85	15.85	17.35
all-achievable	**18.28**	14.68	14.68	**18.28**
centralized		14.65		**19.29**
S_R	pc	pp	cp	cc
best-cost	15.25	**18.61**	15.74	14.18
load-balance	14.76	16.21	16.21	14.76
rest-achievable	12.17	11.23	11.23	12.17
all-achievable	15.26	17.80	17.80	15.26
centralized		18.39		15.11
(R, C) pareto-d.	pc	pp	cp	cc
best-cost	98	97	118	118
load-balance	87	83	82	88
rest-achievable	74	77	74	74
all-achievable	**132**	100	100	131
centralized		115		**148**

Table 5: Summary of results in all domains.

coverage	pc	pp	cp	cc
best-cost	78	75	77	79
load-balance	76	74	74	77
rest-achievable	**80**	**80**	**80**	**80**
all-achievable	73	72	71	73
centralized		74		77
S_T	pc	pp	cp	cc
best-cost	57.64	50.75	58.21	62.23
load-balance	55.99	49.50	50.49	55.11
rest-achievable	**77.11**	67.71	70.06	73.31
all-achievable	51.29	47.03	47.40	50.02
centralized		52.24		61.28
S_C	pc	pp	cp	cc
best-cost	65.09	59.27	64.35	**70.20**
load-balance	64.72	59.23	59.23	65.37
rest-achievable	69.41	66.50	66.50	69.41
all-achievable	65.41	55.76	55.00	65.41
centralized		58.96		**71.31**
S_R	pc	pp	cp	cc
best-cost	54.92	**66.90**	59.20	50.45
load-balance	49.39	58.61	58.61	50.12
rest-achievable	48.04	55.67	55.67	48.04
all-achievable	46.12	62.83	61.91	46.12
centralized		**69.98**		52.29
(R, C) pareto-d.	pc	pp	cp	cc
best-cost	334	338	**380**	337
load-balance	259	297	293	269
rest-achievable	303	323	311	297
all-achievable	267	277	276	263
centralized		431		**436**

limited resources. The objective is to find a plan that maximizes the utility, modeled through goal preferences (possibly keeping the cost under a certain bound (García-Olaya, de la Rosa, and Borrajo 2011)). Keyder and Geffner showed that soft goals can be compiled away avoiding the need to devise specific algorithms for handling them (Keyder and Geffner 2009). Unlike oversubscription planning, our work assumes all goals are hard, every one must be achieved. There is some relation, though, given that one could consider that in our case, we could define as soft goals the fact that each agent achieves each goal. But, then, we would also need to specify that all goals are achieved.

Conclusions and Future Work

We have described an approach that deals with the task of Multi-Agent Planning with agents preferences over goals. We describe how to model those preferences as a planning metric to be used by state-of-the-art planners that can only minimize plan costs. Then, the approach divides planning in two main steps: assignment of public goals to agents and planning. Each of these two steps can be configured to take into account either actions costs, or agents preferences modeled as penalties. We show results in several IPC domains with a set of configurations. As expected, results show that approaches that use cost as the main metric when planning are better than those which use penalties when we want to obtain solutions of better quality in terms of cost. The opposite applies when we want to obtain better rewards; then we must employ rewards (translated into penalties) in the planning step. The approaches with more coverage are those which employ the rest-achievable goal selection, and they are the fastest too, because the goals are well distributed and the division of goals does not overload the agents planning iterations. The pareto-dominance depends on the domain structure; in some domains the approaches using costs to plan are better than the others and the opposite applies in other domains.

As future work, we plan to improve the quality of the solutions using an anytime scheme and, later, to move to optimal planning, considering a pareto-optimal search. Also, we want to implement the soft goals compilation defined in (Keyder and Geffner 2009). Using this compilation, the goals of the original problem would remain as hard goals and preferences agents have over goals would be modeled as soft goals.

Acknowledgments

This work has been partially supported by Spanish MICINN project TIN2011-27652-C03-02.

References

Arregui, J. P.; Tejo, J. A.; Linares-López, C.; and Borrajo, D. 2012. Steps towards an operational sensors network planning for space surveillance. In *Proceedings of the SpaceOps'12*.

Borrajo, D. 2013a. Multi-agent planning by plan reuse. Extended abstract. In *Proceedings of the AAMAS'13*, 1141–1142.

Borrajo, D. 2013b. Plan sharing for multi-agent planning. In Nissim, R.; Kovacs, D. L.; and Brafman, R., eds., *Preprints of the ICAPS'13 DMAP Workshop on Distributed and Multi-Agent Planning*, 57–65.

Brafman, R. I.; Domshlak, C.; Engel, Y.; and Tennenholtz, M. 2009. Planning games. In *Proceedings of IJCAI*, 73–78.

Castillo, L.; Armengol, E.; Onaindía, E.; Sebastiá, L.; González-Boticario, J.; Rodríguez, A.; Fernández, S.; Arias, J. D.; and Borrajo, D. 2008. SAMAP. A user-oriented adaptive system for planning tourist visits. *Expert Systems with Applications* 34(2):1318–1332. ISSN: 0957-4174.

Cenamor, I.; de la Rosa, T.; and Borrajo, D. 2013. Ondroad planner: Building tourist plans using traveling social network information. In *Proceedings of Conference on Human Computation & Crowdsourcing (HCOMP'13). Works-in-Progress & Demonstrations*.

Crosby, M., and Rovatsos, M. 2011. Heuristic multiagent planning with self-interested agents. In *Proceedings of AAMAS*, 1213–1214.

García, J.; Florez, J. E.; Álvaro Torralba; Borrajo, D.; Linares-López, C.; Ángel García-Olaya; and Sáenz, J. 2013. Combining linear programming and automated planning to solve multimodal transportation problems. *European Journal of Operations Research* 227:216–226.

García-Olaya, A.; de la Rosa, T.; and Borrajo, D. 2011. Using relaxed plan heuristic to select goals in oversubscription planning problems. In *Advances in Artificial Intelligence*, volume 7023/2011 of *Lecture Notes on Computer Science*, 183–192. Springer Verlag.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In Shlomo Zilberstein, J. K., and Koenig, S., eds., *Proceedings of ICAPS'04*, 161–170.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Keyder, E., and Geffner, H. 2009. Soft goals can be compiled away. *Journal of Artificial Intelligence Research* 36(1):547–556.

Nissim, R., and Brafman, R. I. 2013. Cost-optimal planning by self-interested agents. In *Proceedings of AAAI'13*.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *JAIR* 39:127–177.

Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *Proceedings of ICAPS'04*, 393–401.