Knowledge-Based Systems 29 (2012) 54–62

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

μ-SATPLAN: Multi-agent planning as satisfiability

Yannis Dimopoulos [a,*], Muhammad Adnan Hashmi [b], Pavlos Moraitis [c]

[a] Department of Computer Science, University of Cyprus, Nicosia, Cyprus
[b] LIP6, Pierre and Marie Curie University, Paris, France
[c] LIPADE, Paris Descartes University, Paris, France

ARTICLE INFO

Article history:
Available online 9 August 2011

Keywords:
Multi-agent planning
Planning as satisfiability
Coordination
Cooperation
Multi-agent systems

ABSTRACT

Planning is a fundamental issue in multi-agent systems. In this work we focus on the coordination of multiple agents in two different settings. In the first, agents are able to attain individual goals that are necessary for the achievement of a global common goal. As the agents share the same environment, they need to find a coordinated course of action that avoids harmful (or negative) interactions, and benefit from positive interactions, whenever this is possible. In the second setting some of the agents may need the assistance of other agents to achieve their individual goals. This is the case where some of the actions of the plan of an agent may be executed by another agent who will play the role of the assistant. We formalize these two problems in a more general way than in previous works, and present a coordination algorithm which generates optimal solutions in the case of two agents. In this algorithm, agents use μ-SATPLAN as the underlying planner for generating individual and joint consistent plans. This planner is an extension of the classical SATPLAN planner, that tackles negative and positive interactions and, therefore, multi-agent planning. We also present an algorithm that solves the assistance problem. The underlying algorithm is again μ-SATPLAN, and is used for the generation of individual (based on assistance) and joint consistent plans. Finally experimental results on multi-agent versions of problems taken from International Planning Competitions demonstrate the effectiveness of μ-SATPLAN and the coordination algorithm.

1. Introduction

Planning is a central issue in multi-agent systems. Several works have been proposed in the literature, covering different aspects of the problem of coordinating the plans of several agents operating in the same environment (see e.g. [1–14]).

In this paper we study the coordination of multiple agents in two different settings. In the first setting agents are able to achieve individual goals by themselves and without any support or assistance by other agents. These individual goals are necessary for the achievement of a global common goal. The agents have complementary capabilities and none of them has the necessary capabilities in order to attain all the goals, and therefore the global goal, alone. In the second setting the agents have still to achieve individual goals for the achievement of a common goal, but now some of them need the assistance of other agents in order to attain their individual goals. Therefore, some of the actions of their plans will be executed by other agents, who will play the role of assistants.

We assume that in both settings the failure of individual goals and therefore the failure of the global goal, is worst than achieving individual and global goals through suboptimal plans (i.e. plans with greater cost or execution time). Therefore suboptimal plans may be taken in consideration by the agents during the planning process. Both settings belong to the more general case of multi-agent planning where both *planning* and *execution* are *distributed* among the different agents.

As the agents operate in the same environment conflicts (e.g. due to the sharing of resources or the interdependency of actions) may arise. Therefore, they need to coordinate their course of action in order to avoid *harmful (or negative) interactions* and also to benefit from situations where *positive interactions* may arise [2]. We call the first setting *multi-agent coordinated actions problem* (MACAP). Moreover, there can be more than one solution to a MACAP, and we are interested in finding an optimal one. We call the problem of finding an optimal solution *multi-agent optimal coordinated actions problem* (MAOCAP). We call the second setting *multi-agent assistance actions problem* (MAAAP).

On a high level, an important difference of our approach from other works, is that it handles both coordination and assistance in a uniform way. Moreover, the use of a classical method, brings

* Corresponding author.
E-mail addresses: yannis@cs.ucy.ac.cy (Y. Dimopoulos), adnan.hashmi@lip6.fr (M.A. Hashmi), pavlos@mi.parisdescartes.fr (P. Moraitis).

0950-7051/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2011.07.019

recent and future advances in classical planning into multi-agent planning. On a more technical level, and as far as coordination is concerned, the works closer to ours are [10,4]. Although there are similarities to the techniques presented in [10], our approach tackles a more general problem. The main difference is that the input to the algorithm of [10] is a fixed combination of individual plans that need to be coordinated, whereas our coordination procedure generates all such possible combinations from which it selects an optimal one. On the other hand, the problem we tackle here bares a resemblance to the one addressed in [4]. However, our solution to the problem presents certain advantages. Firstly, our algorithm is able to find optimal solutions, which is not the case for [4] when there are "positive" interactions between the actions. Secondly, if optimality is not required, our algorithm can terminate as soon as a consistent solution of acceptable quality is found, or a time limit is reached, which is not the case for [4], where only the optimal solution (assuming no positive interactions between actions) is returned when the process terminates.

This work is based on the ideas of [15] which it significantly extends. More precisely, compared to the above work, this paper proposes μ-SATPLAN, an extension of the well-known classical planner SATPLAN to deal with the multi-agent planning problem by handling negative as well as positive interactions. The present work also presents experimental results on multi-agent versions of planning problems taken from classical planning. These results show that our method is a viable approach to multi-agent planning. In a few words, our approach is an extension of techniques similar to those presented in [10] to the more general problem studied in [4], with the additional advantage of an efficient underlying classical planner.

It is worth noting that the idea of using satisfiability for solving the closely related problem of plan merging has been considered in [16]. However, in plan merging no new actions can be added in the new plan, i.e. an action cannot belong to the final plan if it does not appear in some of the plans that are merged. Moreover, [16] does not study the parallel encoding of planning into satisfiability that we investigate here. The idea of distributed planning as SAT problem has also been discussed in [17] where the distributed planning problem is seen as the process of finding a satisfying assignment for both propositional encodings performed by each agent, given the domain theory, the goal to be reached and the initial situation he is interested in, that is consistent on variables that are shared between agents. Shared variables are identified as those variables related to the same event of fact in the knowledge base of both agents. However this work proposes only a general description of how a distributed planning problem could be considered as a distributed satisfiability problem without any detail about the way agents are generating individual plans, the way negative and positive interactions are handled, or individual plans are exchanged and joint plans are generated and of course without an implemented system and experimental results.

Finally, the Distributed Graphplan algorithm [18] extends the classical Graphplan method [19] to a multi-agent setting. The algorithm essentially solves a plan merging problem i.e., an action cannot be included in the joint plan if it is not in the plan of at least one of the individual agents. Moreover, unlike μ-SATPLAN, Distributed Graphplan does not tackle positive interactions, and is restricted to two agents.

The rest of the paper is organized as follows. Section 2 recalls some background for the planning as satisfiability framework, whereas Section 3 defines formally MACAP and MAOCAP. Section 4 presents the solution to MACAP by presenting μ-SATPLAN, our extension of the SATPLAN to multiple agents. Sections 5 presents the proposed solution for MAOCAP. Section 6 presents the MAAAP setting, while Section 7 presents the experimental results for the MACAP setting. Section 8 concludes and discusses future work.

2. Propositional satisfiability based planning

We assume that the agents' planning domain theories are described in the STRIPS language, and denoted by D_α the set of actions that appear in the domain theory of agent α. To generate their plans the agents use μ-SATPLAN, our extension of SATPLAN system [20]. The rationale behind choosing the propositional satisfiability approach to planning is twofold. First, it is one of the most computationally effective approaches to optimal (wrt plan length) STRIPS planning [21,22]. Second, it can be easily extended to accommodate the needs of the multi-agent planning scenario, but also other desirable features such as planning with preferences (e.g. [23,24]).

We assume that the reader is familiar with the propositional satisfiability encoding of STRIPS planning. Here we recall very briefly the basics of SATPLAN approach to planning. First, a plan length k is assumed, and the planning problem is translated into a propositional theory (set of clauses). If the resulting satisfiability problem is solvable, a plan of length k is extracted from the model that has been found. Otherwise, the plan length is set to $k+1$ and the procedure iterates.

Among the several ways to transform a planning problem into satisfiability one, we adopt the Graphplan-based parallel encoding [20]. The facts of the (fully specified) initial state and the final state are translated into literals. The propositional theory also contains clauses that constraint actions to imply their preconditions, and fluents to imply the disjunction of all actions that have these fluents in their add-effects. Finally, conflicting actions are declared as mutual exclusive through suitable binary clauses that are added to the theory. For a description of the latest version of SATPLAN the reader is referred to [22], and for an introduction to planning as satisfiability to [25].

In the following we assume that a plan is a set of temporal propositions of the form $A(t)$, where A is an action and t is a time point, meaning that action A executes at time t. If D is a domain theory, I an initial state, P a plan and G a set of goals, the notation $P \models_{D,I} G$ denotes that P is a plan for goal G in the domain D with initial state I, under the standard STRIPS semantics. When there is no possibility for confusion, we simply write $P \models G$.

For the purposes of multi-agent assistance actions problem we slightly extend the STRIPS language to accommodate the representational needs of cooperation between the agents. In the extended language, the preconditions $prec(A)$ of an action A is the union of the sets $norm\_prec(A)$ and $extern\_prec(A)$. The set $norm\_prec(A)$ contains normal action preconditions, i.e. fluents that must hold before the execution of the action for that action to succeed. The elements of the set $extern\_prec(A)$ are fluents that the agent may request some other agent to bring about in the world. Therefore, the agent can assume that these fluents will be true in the world when needed, and ignore them during planning. In the context of the SATPLAN framework this means that the agent plans by taking into account only the propositions in the sets $norm\_prec$ and ignores $extern\_prec$. The following is an example of a multi-agent assistance actions problem.

Example 1. Let D_A be the domain theory of agent A that includes the operators $pickup(X,L)$ and $putdown(X,L)$ for picking up and putting down an object X at location L, respectively, with the usual preconditions and effects. Moreover, D_A contains the operator $move(X,Y)$ for moving from location X to location Y, with preconditions $norm\_prec(move(X,Y)) = \{at(X,Y)\}$ and $extern\_prec(move(X,Y)) = \{opendoor(X,Y)\}$, and the usual effects. The external precondition $opendoor(X,Y)$ means that the agent expects that some other agent will open the doors for him. Assume the initial state $I_A = \{at(l1), at(obj,l1)\}$ and the set of goals $G_A = \{at(obj,l2)\}$. A plan for this problem

is $P_A = \langle\{pickup(obj,l1,0), move(l1,l2,1), putdown(obj,l2,2)\},\{opendoor(l1,l2,1)\}\rangle$. Note that for this plan to succeed, some other agent needs to bring about $opendoor(l1,l2)$ at time 1.

The domain theory of an agent may contain different versions of the same action A, say A_1 and A_2, that differ only in the elements they contain in their sets *norm_prec* and *extern_prec*, i.e. $prec(A_1) = prec(A_2)$ but $norm\_prec(A_1) \neq norm\_prec(A_2)$.

3. MACAP and MAOCAP

In a multi-agent coordinated actions scenario, a number of agents need to generate individual plans that achieve individual goals which are necessary for the achievement of a common global goal. The case is illustrated by the following example.

Assume two agents α and β, and two object, one heavy and one fragile, that lay on a table. The common goal of the agents is to clear the table (i.e. put the objects on the floor). Agent α can move heavy objects and agent β fragile ones. The objects can be moved by the two agents in any order or in parallel. Thus the goal of the agent α is to put the heavy object on the floor and the goal of agent β to put the fragile on the floor. It is therefore obvious that none of the agents has is able to achieve the common goal (i.e. to clear the table), alone and its attainment depends on the cooperation of agents α and β.

We restrict ourselves to the case of two agents, and study a scenario that is defined by the following characteristics.

- Each agent is able to achieve his goals by himself. These individual goals may be necessary for the achievement of a global common goal that none of the agents can achieve alone. Moreover, agents have different capabilities. In the simplest case, the effects of the actions of the agents are disjoint.
- Plan length is the criterion for evaluating the quality of both the individual and the joint plans, with preference given to the joint plan length.

The *coordinated actions* problem is defined formally as follows:

Definition 1 (*MACAP*). Given two agents α and β with goals G_α and G_β that are necessary for a global common goal G_{global} achievement i.e. $G_{global} = G_\alpha \cup G_\beta$, initial states I_α and I_β, and sets of actions D_α and D_β, respectively. Find a pair of plans (P_α, P_β) such that

- $P_\alpha \models_{D_\alpha, I_\alpha} G_\alpha$ and $P_\beta \models_{D_\beta, I_\beta} G_\beta$
- P_α and P_β are non-conflicting

Such pair of plans (P_α, P_β) is called a solution to the MACAP.

We refer to the plans P_α and P_β as *individual* plans, and to the pair (P_α, P_β) as *joint* plan. Moreover, we use the term joint plan to also refer to the plan $P_\alpha \cup P_\beta$. The length of a joint plan (P_α, P_β) is defined as $max(l(P_\alpha), l(P_\beta))$.

Definition 2 (*MAOCAP*). Given two agents α and β with goals G_α and G_β that are necessary for a global common goal G_{global} achievement i.e. $G_{global} = G_\alpha \cup G_\beta$, initial states I_α and I_β, and sets of actions D_α and D_β, respectively. Find a pair of plans (P_α, P_β) such that

- (P_α, P_β) is a solution to the MACAP for agents α and β.
- There is no other solution $\left(P'_\alpha, P'_\beta\right)$ to the problem such that $max\left(l(P'_\alpha), l\left(P'_\beta\right)\right) < max(l(P_\alpha), l(P_\beta))$.

In MAOCAP, agents seek to minimize the length of the joint plan, even in the case where this leads to non-optimal individual plans.

4. Solving the MACAP using μ-SATPLAN

MACAP is solved in a setting where an agent, say agent α computes his individual plan without taking into account possible conflicts with the plans of other agents. Then this plan is sent to the other agent, say agent β, who computes a plan that is not in conflict (i.e. with no *negative interactions*) with the plan of agent α, and which avails the cooperative opportunities (i.e. *positive interactions*) offered by agent α, if such opportunities exist. Such a plan of agent β is called consistent with the plan of agent α. *Negative* and *positive* interactions have been suggested by von Martial [2].

The *negative interactions* occur when agents mutually hinder themselves in reaching their goals. They may come from two different sources that are discussed below.

1. *Causal link threatening*. This conflict is well known in the context of partial order planning [26]. Let $A_1(t_1)$ and $A_2(t_2)$ be two actions of a plan P such that $t_1 < t_2$ and $A_1(t_1)$ is the latest action of the plan P_1 that adds the precondition p of action $A_2(t_2)$. Then, we say that there is causal link between time points t_1 and t_2 related to p, denoted by the triple (t_1, t_2, p).
   Furthermore, if p is a precondition of an action $A(t)$, p appears in the initial state, and there is no action in plan P that adds p and is executed at some time point $t' < t$, then there is a causal link $(0, t, p)$ in P. Moreover, if $A(t)$ is the last action that adds a goal g, there exists a causal link (t, t_{fin}, g), where t_{fin} is the plan length.
   Finally, if p is a proposition that belongs both to the initial and the final state of planning problem, and there is no action in plan P that adds p, then P contains the causal link $(0, t_{fin}, p)$.
   An action $A(t)$ *threatens the causal link* (t_1, t_2, p) if $t_1 \leqslant t \leqslant t_2$ and A deletes p.
2. *Parallel actions interference*. This conflict was introduced in Graphplan [19]. Two actions interfere if they are executed in parallel and one deletes the preconditions or add effects of the other.

The *positive interactions* are all those relationships between two plans from which a benefit for at least one party can be derived. Such a positive relationship can be requested (explicit) or non-requested (implicit). von Martial distinguishes three types of non-requested relationships.

Action equality: Two agents plan to perform an identical action. While recognizing this fact they agree that only one agent performs the action and makes the result available to the other.

Consequence: One agent's actions have the side-effect to achieve one of another agents goals as well. So the second one is relieved from pursuing that strategy.

Favor: The plan of one agent has the side-effect to contribute in some way to the goals of another agent.

Thus in our context the agent β can use some effects generated by agent α's actions and therefore avoid the need to attain facts that have been already brought about by agent α.

The following sections explain how agents α and β compute their plans using μ-SATPLAN.

4.1. Independent plan computation

Initially, agent α computes an individual plan along with a set of causal links. This is done by μ-SATPLAN which is invoked by the call $ComputeNewPlan(T_\alpha, G_\alpha, L, P_\alpha, C_{P_\alpha})$, where T_α includes the agent's domain theory and initial state, G_α is the set of goals of the agent and L is an upper bound on the length of the generated plan (i.e. if $l(P_\alpha)$ is the length of the generated plan, $l(P_\alpha) < L$ holds). This call either returns a plan P_α that achieves all goals of G_α or *fail* in argument P_α. This call also returns the set of causal links C_{P_α} of

plan P_α. μ-SATPLAN uses the original SATPLAN to compute plan P_α and computes the set of causal links C_{P_α} using Algorithm 1.

Algorithm 1: Computing Causal Links

```
C_P ← ∅
for Every level i from goal level going back to level 1 do
  for Every action a at level i do
    for Every precondition p of action a do
      Search in previous layers the latest action, which adds fact p
      if Found an action at level k, which adds fact p then
        Add causal link (k,i,p) to the set C_P
      end if
      if No action found which adds fact p then
        Add causal link (0,i,p) to the set C_P
      end if
    end for
  end for
end for
for Every goal fact g do
  for Every level i starting from goal level going back to level 1 do
    for Every action a at level i do
      Search in previous layers the latest action, which adds fact g
      if Found an action at level k which adds fact g then
        Add causal link (k,goal level,g) to the set C_P
      end if
      if No action found which adds fact g then
        Add causal link (0,goal level,g) to the set C_P
      end if
    end for
  end for
end for
```

4.2. Coordinated plan computation

Agent β receives a plan P_α and a set of causal links C_{P_α} from agent α and computes a plan P_β which is consistent with P_α by invoking μ-SATPLAN through the call *ComputeCoordinatedPlan* $(T_\beta, G_\beta, P_\alpha, C_{P_\alpha}, P_\beta)$, where T_β includes the agent's domain theory and initial state, G_β is the set of goals of the agent. The plan generation method involves resolving the problem of negative and positive interactions. The method that is employed for tackling this problem is detailed in the following sections.

4.2.1. Handling negative interactions

As it has been discussed earlier, negative interactions come either from *causal link threatening* or from *parallel actions interference*. This section explains how μ-SATPLAN resolves *causal link threatening*.

During the construction of his planning graph, agent β checks for all operators O, whether O at action level i threatens any of the causal links (t_1, t_2, p) from the set C_{P_α} where $t_1 \leqslant i \leqslant t_2$. If this holds true, agent β does not add operator O in the planning graph at level i, even if the case where all its preconditions are satisfied at this level. As agent β expands his planning graph by adding new levels, he may add O at later levels which do not threaten the causal link. If the threat persists through all levels, operator O is abandoned.

Example 2. Assume the Blocks World domain where $ON(X,Y)$ means that block X is on block Y and $MOVE(X,Y,Z,i)$ represents the action of moving block X from Y onto Z at time i. Furthermore, assume that C_{P_α} contains the causal link $(1,3,ON(A,B))$. Then, $MOVE(A,B,C,1)$ cannot be added to the planning graph of agent β as it threatens a causal link of agent α. The same holds for levels 2 or 3. In fact, causal link $(1,3,ON(A,B))$ means that block A should be on block B from time 1 to time 3, as it is needed by agent α at time 3. If agent β moves block A from B onto C between time points 1 and 3, the plan of agent α is destroyed.

4.2.2. Handling positive interactions and parallel actions interference

Handling positive interactions involves further modifications to the original SATPLAN system, that have been implemented in μ-SATPLAN.

When μ-SATPLAN attempts to generate a plan for agent β, it adds a fresh action, called *NONAME*, for each time step i in the plan of agent α. This means that if there are n time steps in the plan of agent α, it creates n actions *NONAME* namely *NONAME*(0), *NONAME*(1), *NONAME*(2), ..., *NONAME*(n) such that:

- The add effects of action *NONAME*(i) are the facts added by (the actions of) agent α at time i.
- The delete effects of action *NONAME*(i) are the facts deleted by agent α at time i.
- The preconditions of action *NONAME*(i) are the facts that are preconditions of the actions of agent α at time i.

Thus an action *NONAME*(i) represents all the actions in the plan of agent α, which are executed at time i. Agent β while constructing his planning graph, explicitly puts the action *NONAME*(i) at action level i. So it is obvious that proposition level i has now all the facts added or deleted by agent α in his plan at time i. By doing so, agent β has all the information about the facts added and deleted by agent α at each level.

In the SATPLAN algorithm, the planning graph translates into a CNF theory and then a solver searches for a satisfying truth value assignment for this theory. From this truth assignment a solution to the planning problem is extracted. Here an important issue arises. The purpose of adding *NONAME* actions in the planning graph of agent β is to ensure that the agent does not introduce any actions in his plan that are in conflict with the plan of agent α. To achieve this, μ-SATPLAN adds these *NONAME* actions as unary clauses in the CNF theory. Thus the solver now has to find a solution that includes all the *NONAME* actions. This approach has an important advantage. It solves easily the problem of *positive interactions*, and at the same time it also tackles *parallel actions interference* as follows. All actions that interfere with those of α are marked as mutually exclusive (by appropriate binary clauses) in the CNF theory of agent β. Therefore, since the *NONAME* actions must be necessary true in all models of the theory, all interfering actions of agent β are excluded from his plans. The following example illustrates the approach.

Example 3. Assume two agents α and β, and suppose that agent α has already computed his plan $P_\alpha = \{A1(0), A2(0), A3(1)\}$. The add effects of the actions of P_α are $eff(A1) = a0$, $eff(A2) = a1$, $eff(A3) = a2$. Agent α sends this information to agent β. Agent β creates two *NONAME* actions as there are two time steps in the plan of agent α. The add effects of *NONAME*(i) equals the union of all the add effects of the actions of agent α at time i. So we have $eff(NONAME(0)) = eff(A1) \cup eff(A2) = \{a0, a1\}$ and $eff(NONAME(1)) = eff(A3) = \{a2\}$. The domain theory of agent β contains the actions $D_\beta = \{B1, B2, B3, B4\}$ with the following preconditions and add effects: $prec(B1) = \{a6\}$, $eff(B1) = \{a5\}$, $prec(B2) = \{a5\}$, $eff(B2) = \{a0\}$, $prec(B3) = \{a5\}$, $eff(B3) = \{a7\}$, $prec(B4) = \{a0, a7\}$, $eff(B4) = \{a8\}$. The goal of agent β is $G_\beta = \{a2, a4, a7, a8\}$. Thus agent β creates his planning graph and adds all actions *NONAME*(i) at action level i. The planning graph of agent β is shown in Fig. 1. Gray lines are NOOPS, and boxes

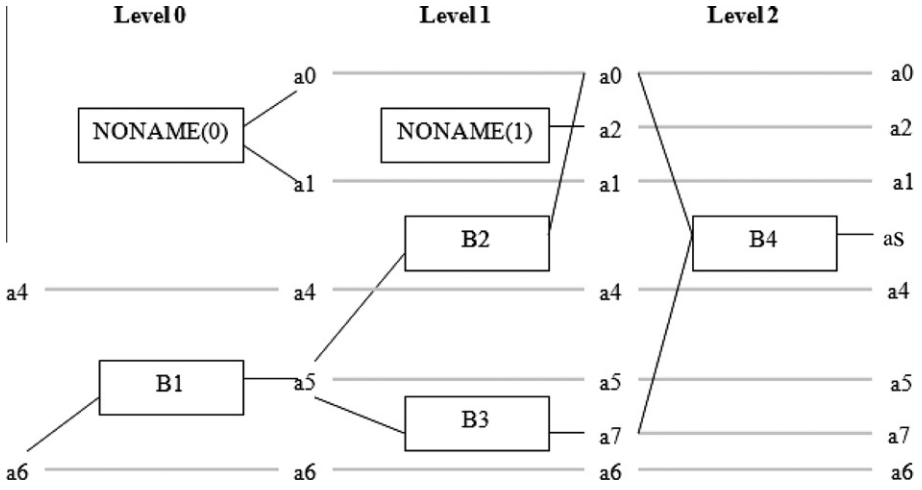

Fig. 1. Illustrating positive interactions.

correspond to actions. Small letters followed by numbers are propositions. A line from proposition F to an action O means that F is the precondition of O. A line from an action O to a proposition F means that F is an add effect of O. Observe that $a0$ is needed by $B4$ to produce $a8$, and there are two actions which add $a0$, namely $NONAME(0)$ and $B2$. So the solver has to choose between $NONAME(0)$ and $B2$, when this planning graph is converted into a CNF theory. As $NONAME(0)$ and $NONAME(1)$ are unit clauses they must be assigned the value true, and therefore the plan of agent β is $P_\beta = \{B1(0), B3(1), B4(2)\}$. Therefore agent β does not re-establish $a0$ which has already been established by agent α.

Finally, we note that μ-SATPLAN is capable of solving MACAP for n agents, with $n > 2$. This means that μ-SATPLAN can be used for finding a plan for the nth agent when $n-1$ agents have already computed their non conflicting plans. Suppose that $n-1$ agents have generated their consistent plans $P_1, P_2, \ldots, P_{n-1}$. Suppose also that the sets of their causal links are $C_1, C_2, \ldots, C_{n-1}$. Then a call $ComputeCoordinatedPlan(T_n, G_n, P_1 \cup P_2 \cup \ldots \cup P_{n-1}, \quad C_1 \cup C_2 \cup \ldots \cup C_{n-1}, P_n)$ returns a plan P_n for the nth agent which is consistent with the plans of $n-1$ agents.

5. Solving the MAOCAP

Algorithm 2 is a coordination procedure that solves MAOCAP for two agents α and β with goals G_α and G_β, domain theories D_α and D_β and initial states I_α and I_β, respectively. Each agent uses μ-SATPLAN for plan generation, and exchanges messages with the other agent.

First agent α computes his plan P_α by invoking μ-SATPLAN via the call *ComputeNewPlan*. Plan P_α is sent to agent β as a candidate sub-plan of a joint plan. Then, agent β computes a plan P_β consistent with P_α by calling procedure *ComputeCoordinatedPlan* of μ-SATPLAN, and sends the joint plan (P_α, P_β) to agent α. At this point (P_α, P_β) becomes the best current joint plan and now it's agent β's turn to compute and propose a candidate sub-plan, which is then processed by agent α. In this way agents take turns to generate and propose candidate sub-plans which are then processed by the other agent who attempts to generate a new consistent joint plan. Each time, a joint plan that is shorter than the current best joint plan is found, it becomes the current best joint plan. In this way the agents split the work of optimal joint plan generation.

The agents exchange messages of the form (P_1, P_2), where P_1 and P_2 are (possibly empty) individual plans. The coordination algorithm (Algorithm 2) refers to agent β and describes how these messages are processed by the agents. The messages can be of three different types, each carrying a different meaning. They are either of the type (P_1, P_2), or $(P_1, \emptyset)$, or $(\emptyset, \emptyset)$, where P_1 and P_2 are non-empty plans. The meaning of each of these messages, and the reaction of the agents to these messages, are described in the following.

Before entering the main body of the algorithm, the agents go through a phase in which the variables and data structures of the algorithm are initialized. Moreover, agent α sends a message of the form $(P, \emptyset)$, where P is the (optimal) plan generated by the call $ComputeNewPlan(T, G, \infty, P, C)$, where T and G are the agent's domain theory and goals respectively.

Each incoming message is processed by the coordination algorithm in a way that depends on its type. A message of the form $(P, \emptyset)$, means that the other agent proposes P as a candidate sub-plan of a joint plan. The set of causal links C is also sent along with the plan. In order to simplify the presentation, it is omitted here. The receiving agent checks, by invoking μ-SATPLAN as explained earlier, if he can generate a plan P' that achieves his own goals and is consistent with P. An additional requirement is that the length of the joint plan $P \cup P'$, defined as $max(l(P), l(P'))$, must be shorter than the best joint plan. If this is the case, the agent sends the message (P, P') to the other agent, meaning that P can be a part of an improved joint plan (P, P'). If the agent that receives the message $(P, \emptyset)$ fails to find a plan as specified above, he sends the message $(P, fail)$, indicating that P cannot be part of a better joint plan. Then, the agent attempts to generate a new sub-plan with length shorter than l_{best}. If such a sub-plan exists, he sends it to the other agent. Otherwise, he sends the message $(\emptyset, \emptyset)$, indicating that there are no shorter individual plans.

A received message of the form (P_1, P_2), with $P_1 \neq \emptyset$ and $P_2 \neq \emptyset$, is a reply to an earlier message, where the agent receiving (P_1, P_2) has proposed plan P_1 to the other agent. Upon processing such a message, if $P_2 \neq fail$ and the proposal (i.e. plan P_2) leads to an improved joint plan (P_1, P_2), the variables P_{best} and l_{best} are updated accordingly. If $P_2 = fail$ then the agent removes P_1 from the set of stored plans as it does not lead to an improved joint plan. This ensures that at any point at most two plans are stored, one which is under consideration and one which is part of the current optimal plan. The agent that receives (P_1, P_2) attempts to generate a new candidate sub-plan which is sent to the other agent. Otherwise a message $(\emptyset, \emptyset)$ is sent.

Upon receiving a message $(\emptyset, \emptyset)$, an agent sets his *expect* variable to false, meaning that he does not expect any further candidate sub-plans from the other agent. If variable *continue* is true, the agent generates another plan that is sent to the other agent. If this is not possible he exits the algorithm.

The algorithm terminates when the condition (*not continue*) $\wedge$ (*not expect*) becomes true. In such a case, the agent has received replies to all the sub-plans that he has proposed, he has no

other plan to propose, and he does not expect any further proposals from the other agent.

Algorithm 2: Coordination Algorithm

while *true* **do**
get_incom_message (P_α,P)
if $P_\alpha \neq \emptyset$ and $P = \emptyset$ **then**
  ComputeCoordinatedPlan($T_\beta, G_\beta, P_\alpha, C_{P_\alpha}, P_\beta$)
  if $P_\beta \neq fail$ and $max(l(P_\alpha), l(P_\beta)) < l_{best}$ **then**
    l_{best}:=$max(l(P_\alpha), l(P_\beta))$, $P_{best} := (P_\alpha, P_\beta)$
    send message (P_α,P_β)
  else
    send message (P_α,*fail*)
  if (*continue*) **then**
    Call Procedure *New_Proposal_Coordination*
  if (*not continue*) and (*not expect*) **then**
    exit (P_{best})
else
if $P_\alpha \neq \emptyset$ and $P \neq \emptyset$ **then**
  if $P \neq fail$ **then**
    $l_{best} := max(l(P_\alpha), l(P))$, $P_{best} := (P_\alpha, P)$
  else
    Delete P from memory
  if *continue* and (*not expect*) **then**
    Call Procedure *New_Proposal_Coordination*
  if (*not continue*) and (*not expect*) **then**
    exit (P_{best})
else (i.e. $P_\alpha = \emptyset \wedge P = \emptyset$)
  if (*not continue*) **then**
    exit (P_{best})
  if (*continue*) **then**
    expect = false
    Call Procedure *New_Proposal_Coordination*
    exit (P_{best})

Procedure 3: *New_Proposal_Coordination*

ComputeNewPlan($T_\beta, G_\beta, l_{best}, P_\beta, C_{P_\beta}$)
if $P_\beta \neq fail$ **then**
  send message (P_β,$\emptyset$)
else
  continue = false
  send message ($\emptyset$,$\emptyset$)

6. Multi-agent assistance actions problem

As in the MACAP case, in MAAAP two agents α and β need to achieve their goals with non-conflicting individual plans. The difference here is that one of the agents may request the other agent to achieve specific subgoals that will enable the requesting agent to attain his own goals. When an agent receives such a request, he attempts to generate a plan that, apart from his own goals, also achieves the requesting agent's subgoals. The MAAAP problem can be defined formally as follows.

Definition 3 (*Multi Agent Assistance Actions Problem (MAAAP)*). Given two agents α and β with goals G_α and G_β, initial states I_α and I_β and sets of actions D_α and D_β, respectively, find a pair of plans (P_α, P_β) such that

- $P_\alpha = P_\alpha^\alpha \cup P_\alpha^\beta$ and $P_\beta = P_\beta^\beta \cup P_\beta^\alpha$;
- $P_\alpha^\alpha \cup P_\beta^\alpha \models_{D_\alpha \cup D_\beta, I_\alpha \cup I_\beta} G_\alpha$ and $P_\beta^\beta \cup P_\alpha^\beta \models_{D_\alpha \cup D_\beta, I_\alpha \cup I_\beta} G_\beta$;
- if $P_\alpha^\alpha \models_{D_\alpha, I_\alpha} G_\alpha$ then $P_\beta^\alpha = \emptyset$, and if $P_\beta^\beta \models_{D_\beta, I_\beta} G_\beta$ then $P_\alpha^\beta = \emptyset$;
- if $c(t) \in P_\alpha$ then $c \in D_\alpha$, and if $c(t) \in P_\beta$ then $c \in D_\beta$;
- P_α and P_β are non-conflicting.

Note that coordinated actions setting is a special case of assistance actions setting, where P_α^β and P_β^α are empty. Also, we may impose the additional requirement that in an assistance actions scenario one or both agents are not able to achieve their goals by themselves, in other words $\neg\exists P$ s.t. $P \models_{D,I} G$, where D, I and G are the domain theory, the initial state and the goal of the agent. However, we do not enforce this restriction. Indeed, it can be the case that an agent can achieve some of his subgoals by executing his own actions, but he may seek the assistance of other agents as this may lead to better quality (in our case shorter) plans.

The assistance procedure for agent β is given in Algorithm 4. It is very similar to the coordination algorithm, with the main differences being in the way the classical planner is invoked and the form of the messages that the agents exchange. The main difference in the implementation of μ-SATPLAN for assistance actions problem is discussed in Section 6.1. In the assistance algorithm agents generate their individual plans via a call to procedure *ComputeNewPlan*($T, G, L, \langle P, R\rangle, C_P$), which given the CNF encoding T of the domain theory (represented in the extended STRIPS language described in Section 2), a set of goals G and a bound l, generates a plan P together with a (possibly empty) set of assistance requests R. The plan P succeeds only if the propositions of the set R are true at their specified time points. This more general form of generated plans necessitates a slightly more complex form of messages. The messages are now of the form ($\langle P, R\rangle, P'$), where P and P' are plans and R is a set of assistance requests. When an agent generates a new individual plan $\langle P, R\rangle$, he sends out the message ($\langle P, R\rangle, \emptyset$). Moreover the agents also send a set of constraints which are omitted here.

Algorithm 4: Assistance Actions Algorithm

while *true* **do**
get_incom_message ($\langle P_\alpha, R_\alpha\rangle$,$P$)
if $P_\alpha \neq \emptyset$ and $P = \emptyset$ **then**
  ComputeAssistanceActionsPlan($T_\beta, G_\beta, P_\alpha, R_\alpha, C_{P_\alpha}, \langle P_\beta, \emptyset\rangle$)
  if $P_\beta \neq fail$ and $max(l(P_\alpha), l(P_\beta)) < l_{best}$ **then**
    $l_{best} := max(l(P_\alpha), l(P_\beta))$, $P_{best} := (P_\alpha, P_\beta)$
    send message ($\langle P_\alpha, R_\alpha\rangle$,$P_\beta$)
  else
    send message ($\langle P_\alpha, R_\alpha\rangle$,*fail*)
  if (*continue*) **then**
    Call Procedure *New_Proposal_Assistance*
  if (*not continue*) and (*not expect*) **then**
    exit (P_{best})
else
if $P_\alpha \neq \emptyset$ and $P \neq \emptyset$ **then**
  if $P \neq fail$ **then**
    $l_{best} := max(l(P_\alpha), l(P))$, $P_{best} := (P_\alpha, P)$
  else
    Delete P from memory
  if *continue* and (*not expect*) **then**
    Call Procedure *New_Proposal_Assistance*
  if (*not continue*) and (*not expect*) **then**
    exit (P_{best})
else (i.e. $P_\alpha = \emptyset \wedge P = \emptyset$)
  if (*not continue*) **then**
    exit (P_{best})
  if (*continue*) **then**
    expect = false
    Call Procedure *New_Proposal_Assistance*
    exit(P_{best})

As in the coordination case, each incoming message is processed by the assistance algorithm of the receiving agent in a way that depends on its type. A message of the form $(\langle P_\alpha, R_\alpha\rangle, \emptyset)$ in the incoming message queue of agent β, is interpreted as a request to search for a plan that is consistent with P_α and achieves R_α. Upon processing such a message, agent β invokes μ-SATPLAN via the call *ComputeAssistancePlan*$(T_\beta, G_\beta, P_\alpha, R_\alpha, C_{P_\alpha}, \langle P_\beta, \emptyset\rangle)$. The empty set in the last parameter $\langle P_\beta, \emptyset\rangle$ of the call, enforces the generation of a plan that does not contain assistance requests. This means that agent β must achieve his goals without the assistance of agent α. In a more general version of the assistance algorithm, one could allow a call of the form *ComputeAssistancePlan*$(T_\beta, G_\beta, P_\alpha, R_\alpha, C_{P_\alpha}, \langle P_\beta, R_\beta\rangle)$, where agent β may reply to agent α with a plan P_β but also a set R_β of assistance requests. In the general case, we may end up with a situation of "nested assistance", where an agent can reply to an assistance request with a new assistance request. Such a situation terminates successfully if one of the agents achieves his goals without the need for further assistance. However, handling the general case requires more complicated data structures, and it is not discussed further.

All other message types are processed by the assistance procedure in a manner similar to the way they are handled by the coordination algorithm.

6.1. Handling cooperation

The handling of cooperation is implemented in μ-SATPLAN by a number of additional modifications to the original SATPLAN algorithm.

Agent α computes his plan by taking into account only the normal preconditions of the actions and ignores all external preconditions. Therefore, if all the normal preconditions of an action A are valid at time step i then α may add this action in his plan even if action A's external preconditions are not valid at time step i. This is because agent α plans under the assumption that agent β will add the external preconditions.

For agent β,μ-SATPLAN creates an action *REQUESTED_GOAL*(i) for each time step i in the plan of agent α that includes an action having external preconditions which are assumed to be handled by agent β. The preconditions of *REQUESTED_GOAL*(i) is the union of the external preconditions of all actions that are included at time step i in the plan of agent α, whereas its add and delete effects are empty. It means that, if in the plan of agent α, there are two actions at time steps 2 and 5, which has external preconditions then agent β creates two actions namely *REQUESTED_GOAL*(2) and *REQUESTED_GOAL*(5), such that the preconditions of *REQUESTED_GOAL*(2) and *REQUESTED_GOAL*(5) are the external preconditions of the actions at time steps 2 and 5, respectively in the plan of agent α. Thus an action *REQUESTED_GOAL*(i) represents all the preconditions at time step i in the plan of agent α that has to be fulfilled by agent β. Agent β adds action *REQUESTED_GOAL*(i) at action level i of his planning graph. Moreover, when this planning graph is encoded into a CNF theory, all the *REQUESTED_GOAL* actions are added as unit clauses into the theory. Therefore, all solutions that are returned for agent β by the solver establish the external preconditions of the actions in the plan of agent α.

Procedure 5: *New_Proposal_Assistance*

ComputeNewPlan$(T_\beta, G_\beta, l_{best}, \langle P_\beta, R_\beta\rangle, C_{P_\beta})$
if $P_\beta \neq fail$ **then**
  send message $(\langle P_\beta, R_\beta\rangle, \emptyset)$
else
  continue = false
  send message $(\langle\emptyset, \emptyset\rangle, \emptyset)$

7. Experimental results

In this section we present some preliminary experimental results for the coordination algorithm presented in Section 5. Although the algorithm is able to find optimal solutions to MACAP, doing so can be inefficient. This is because finding an optimal plan (and more importantly, proving optimality) requires searching the entire space of plan combinations, which can be very large even for moderately sized problems. Hence, the current implementation of μ-SATPLAN follows a different approach that aims at generating quickly sub-optimal plans of good quality.

Every time an agent searches for a new candidate sub-plan, the length of the solution that is sought is increased by one. For instance, if the first plan found by an agent is of length 6, the length of his second plan must be 7. This process iterates until the length of the plan reaches the current value of *lbest*. At this point, the value of variable *continue* is set to false. Therefore, the current implementation of the coordination algorithm does not guarantee the optimality of the solutions that it returns, but it has been observed experimentally that the generated solutions are of good quality. In many cases the length of the plan that is returned is considerably shorter than that of the first solution that is found.

We run μ-SATPLAN and the coordination algorithm on *multi-agent versions* of the well-known Logistics planning domain as well as the Storage and TPP domains from the 5th International Planning Competition [21]. The TPP (Traveling Purchaser Problem) is a generalization of the Traveling Salesman Problem. A purchaser can buy products from different markets that provide the products in different amounts and prices. The storage domain is about moving a certain number of crates from some containers to some depots by hoists. Inside a depot, each hoist can move according to a specified spatial map connecting different areas of the depot. For more information about the domains the reader is referred to [21]. To obtain the multi-agent version of a problem, we split the goals of the original problem into different sets and assign each goal set to a different agent. All our experiments concern the case of two agents. We assume that each agent can execute all the actions in the domain, and therefore he can achieve its goals without assistance from other agents.

The results are shown in Table 1. The underlying planning system that is used is SATPLAN 2006. All experiments were run on a machine with a 2.80 GHz CPU, 4096 MBs of memory, and the CPU cut-off limit for μ-SATPLAN was set to 3600 s. Column *Goals* α/β contains pairs of the form a/b where $a(b)$ is the number of goals assigned to agent $\alpha(\beta)$. The columns *Time 1st Plan* and *Length 1st Plan* provide information about the run time (in seconds) and length of the first joint plan found by μ-SATPLAN. More specifically, an entry (a,b) means that in the first joint plan that is found, agent $\alpha(\beta)$ finds a plan of length $a(b)$. Similar information is provided in columns *Best Plan Length* and *Time for Best Plan*, but for the best plan found by the system. Finally, the entries under ♯*Joint Plans* are the total number of joint plans computed by the coordination algorithm, before its termination. An entry *No Plan* in the column *Length 1st Plan* means that there was no consistent plan of agent β for the first plan proposed by agent α.

To understand the effects of positive interactions we discuss a pair of plans generated by μ-SATPLAN (Table 2). The problem under consideration is from TPP domain. There is one market M1 and one depot D1. Moreover there are two trucks T1 and T2 in the world. M1 is selling products P1 and P2. The goal of agent α is to buy and store P1 in D1 and the goal of agent β is to buy and store P2 in D1. We can see from the plan generated by agent β that he does not utilize truck T2, but instead he avails the cooperative opportunities offered by agent α by using the same truck T1. At time 0, agent β remains idle waiting for agent α to move T1 to

Table 1
μ-SATPLAN performance on multi-agent problems.

Problem	Goals α/β	Time 1st plan	Length 1st plan	Best plan length	♯ Joint plans	Time for best plan
AIPSLog11	5/4	3	(9,14)	(13,10)	7	16
AIPSLog15	4/7	4	(10,17)	(15,10)	10	28
AIPSLog18	4/4	5	(10,14)	(11,11)	3	12
AIPSLog20	5/5	7	(14,18)	(14,18)	8	39
AIPSLog24	5/4	3	(12,14)	(13,12)	5	14
AIPSLog28	5/5	7	(10,16)	(15,13)	7	36
AIPSLog30	4/5	4	(12,16)	(12,16)	10	28
Storage10	2/2	12	No plan	(11,18)	16	64
Storage12	2/2	32	No plan	(9,9)	10	216
Storage16	3/3	129	No plan	(13,8)	16	942
TPP11	3/3	4	(13,13)	(13,13)	3	9
TPP13	4/4	7	(9,14)	(10,11)	5	22
TPP14	4/5	8	(7,13)	(9,10)	4	34
TPP15	5/5	12	(9,15)	(11,11)	6	50
TPP17	6/6	29	(11,11)	(11,11)	3	65
TPP19	7/7	47	(10,12)	(11,10)	2	90
TPP20	9/6	58	(12,11)	(12,11)	4	117

Table 2
A plans pair generated by μ-SATPLAN.

Time	Plan of agent α	Plan of agent β
0	(DRIVE T1 D1 M1)	[]
1	(BUY T1 P1 M1)	(BUY T1 P2 M1)
2	(LOAD P1 T1 M1)	(LOAD P2 T1 M1)
3	(DRIVE T1 M1 D1)	[]
4	(UNLOAD P1 T1 D1)	(UNLOAD P2 T1 D1)

M1. At times 1 and 2, agent β also buys and puts his product P2 in T1 along with agent α. Then again at time 3 agent β remains idle waiting for agent α to drive T1 from M1 to D1. At time 4, both agents unload and store their products in D1.

The preliminary experimental results of Table 1 show that μ-SATPLAN represents a viable approach to the problem of multi-agent planning.

8. Conclusion and future work

In this paper we formalized the multi-agent coordinated actions problem (MACAP), multi-agent optimal coordinated actions problem (MAOCAP) and the multi-agents assistance actions problem (MAAAP). We presented μ-SATPLAN, a multi-agent version of SATPLAN the most powerful planner in classical planning, which is used by the agents to solve the MACAP, MAOCAP and MAAAP. The multi-agent assistance algorithm we propose is an innovative approach to dealing with the problem of mutual assistance among agents with complementary capabilities, whereas our coordination procedure presents certain advantages over previous approaches. We also presented all the details of how μ-SATPLAN deals with negative as well as positive interactions in order to find consistent plans for multiple agents working in the same environment. Moreover, we presented for the first time in multi-agent planning domain, several experimental results that show the added value of adapting SATPLAN to multi-agent planning. We believe that presenting these results is an important issue because it will give the opportunity to other researchers working in multi-agent planning to use these domains and to compare the performance of their planners with μ-SATPLAN.

There are several lines for future work. As in our current implementation, we can not guarantee the global optimal solution, so currently we are investigating the use of heuristics to guide the search of the coordination algorithm, in order to ensure the optimality of the global joint plan and at the same time not to generate a lot of sub-optimal plans before coming up with the optimal one. There seems also to be room for improvement by combining the classical planning based algorithms we presented with techniques from multi-agent plan merging as those presented e.g. in [7,10]. Another direction of future study concerns the extension of the proposed framework to other problems from multi-agent planning, such as action synchronization and interleaved planning and execution. As a concluding remark, we reiterate that this work can be seen as a first step towards establishing a framework where different multi-agent planning problems can be studied in the light of recent advances in classical planning.

References

[1] G. Boutilier, R. Brafman, Partial-order planning with concurrent interacting actions, Journal of Artificial Intelligence Research 14 (2001) 105–136.
[2] F. von Martial, Coordinating Plans for Autonomous Agents, vol. 610, Springer Verlag, 1992.
[3] M.J. Katz, J.S. Rosenschein, The generation and execution of plans for multiple agents, Computers and Artificial Intelligence 12 (1) (1993) 5–35.
[4] E. Ephrati, J. Rosenschein, Divide and conquer in multi-agent planning, in: AAAI94, 1994, pp. 375–380.
[5] B.J. Clement, E.H. Durfee, Top-down search for coordinating the hierarchical plans of multiple agents, in: Agents, 1999, pp. 252–259.
[6] J.S. Cox, E.H. Durfee, Efficient and distributable methods for solving the multiagent plan coordination problem, Multiagent and Grid Systems 5 (4) (2009) 373–408.
[7] I. Tsamardinos, M.E. Pollack, J.F. Horty, Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches, in: AIPS'00, 2000, pp. 264–272.
[8] H. Tonino, A. Bos, M.D. Weerdt, C. Witteveen, Plan coordination by revision in collective agent-based systems, Artificial Intelligence 142 (2002) 121–145.
[9] J.S. Cox, E.H. Durfee, Discovering and exploiting synergy between hierarchical planning agents, in: AAMAS, 2003, pp. 281–288.
[10] J. Cox, E. Durfee, An efficient algorithm for multiagent plan coordination, in: AAMAS05, 2005, pp. 828–835.
[11] J.R. Steenhuisen, C. Witteveen, A. ter Mors, J. Valk, Framework and complexity results for coordinating non-cooperative planning agents, in: 4th German Conference on Multiagent System Technologies, (MATES'06), 2006, pp. 98–109.
[12] B. Ottens, B. Faltings, Coordinating agent plans through distributed constraint optimization, in: 18th International Conference On Automated Planning And Scheduling, ICAPS, 2008.
[13] R.I. Brafman, C. Domshlak, From one to many: planning for loosely coupled multi-agent systems, in: ICAPS, 2008, pp. 28–35.
[14] R. Nissim, R.I. Brafman, C. Domshlak, A general, fully distributed multi-agent planning algorithm, in: AAMAS'10, 2010, pp. 1323–1330.
[15] Y. Dimopoulos, P. Moraitis, Multi-agent coordination and cooperation through classical planning, in: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT), 2006, pp. 398–402.
[16] A. Mali, Plan merging and plan reuse as satisfiability, in: 5th European Conference on Planning, ECP99, 1999, pp. 84–96.
[17] M. Benedetti, L.C. Aiello, Sat-based cooperative planning: a proposal, in: Mechanizing Mathematical Reasoning, 2005, pp. 494–513.
[18] M. Iwen, A. Mali, Distributed graphplan, in: Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI2002), 2002, pp. 138–145.

[19] A. Blum, M. Furst, Fast planning through planning graph analysis, Artificial Intelligence Journal 90 (1–2) (1997) 281–300.
[20] H. Kautz, D. McAllester, B. Selman, Encoding plans in propositional logic, in: Principles of Knowledge Representation and Reasoning, KR'96, 1996.
[21] A. Gerevini, P. Haslum, D. Long, A. Saetti, Y. Dimopoulos, Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners, Artificial Intelligence 173 (5-6) (2009) 619–668.
[22] H. Kautz, B. Selman, J. Hoffmann, SATPLAN: planning as satisfiability, in: Booklet of the 2006 International Planning Competition, 2006. <http://zeus.ing.unibs.it/ipc-5/publication.html>.
[23] E. Giunchiglia, M. Maratea, Planning as satisfiability with preferences, in: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, AAAI'07, 2007, pp. 987–992.
[24] E. Giunchiglia, M. Maratea, Introducing preferences in planning as satisfiability, Journal of Logic and Computation 21 (2) (2011) 205–229.
[25] S.J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2010.
[26] D. Weld, An introduction to least commitment planning, AI Magazine 15 (4) (1994) 27–61.