Extending SATPLAN to Multiple Agents

Yannis Dimopoulos, Muhammad Adnan Hashmi, Pavlos Moraitis

Abstract Multi-agent planning is a core issue in the multi-agent systems field. In this work we focus on the coordination of multiple agents in a setting where agents are able to achieve individual goals that may be either independent, or necessary for the achievement of a global common goal. The agents are able to generate individual plans in order to achieve their own goals, but, as they share the same environment, they need to find a coordinated course of action that avoids harmful (or negative) interactions, and benefits from positive interactions, whenever this is possible. Moreover, agents are interested in finding plans with optimal length where preference is given to the length of the joint plan. We formalize these problems in a more general way with respect to previous works and present a coordination algorithm which provides the optimal solution 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 well known classical planner SATPLAN, aiming to deal with negative and positive interactions and, therefore, with multi-agent planning problem. Finally we present the experimental results using the multi-agent planning problems from the domains proposed and used in classical planning, which demonstrate the effectiveness of μ-SATPLAN and the coordination algorithm.

1 Introduction

Multi-agent planning is an important issue in the multi-agent systems field. Several works have been proposed in the literature covering different aspects of the problem

Yannis Dimopoulos
University of Cyprus, CY-1678 Nicosia, Cyprus

Muhammad Adnan Hashmi
University Pierre and Marie Curie, 75005 Paris, France

Pavlos Moraitis
University Paris Descartes, 75270 Cedex 06 Paris, France

of coordinating the plans of several agents operating in the same environment (see e.g. [1], [2], [3], [4], [5], [6], [7], [8],[9]).

In this paper we study the coordination of multiple agents in a setting where agents are able to achieve individual goals that may be either independent (case of self-interested agents), or necessary for the achievement of a global common goal (case of collaborative agents). In the second case agents have complementary capabilities. It means that they can achieve alone the individual goals which are necessary for the achievement of the global goal which is common to both agents. However none of them has the necessary capabilities to achieve all the goals and therefore the global goal alone.

We assume that in both cases the failure of individual goals (and of the global goal if any) is a worse outcome compared to the achievement of individual goals (and global goal) through suboptimal plans (i.e. plans with greater cost or execution time) and, therefore, suboptimal plans may be considered for adoption by the agents during the planning process. The agents are able to generate and execute their plans independently. However, as they operate in the same environment, conflicts 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* might arise. We will call the problem of finding such a pair of plans as *Multi-Agent Coordinated Actions Problem* (MACAP). Moreover, there could be more than one solution to a MACAP and we are interested in finding the optimal one. We will call the problem of finding an optimal solution as *Multi-Agent Optimal Coordinated Actions Problem* (MAOCAP).

Our work extends significantly the work of [10]. More precisely compared to the above work, our work proposes μ-SATPLAN, an extension of famous classical planner SATPLAN to deal with the multi-agent planning problem by handling negative as well as positive interactions. Our work also presents many experimental results using the multi-agent planning problems from the domains proposed and used in classical planning. The experimental results show that our approach is a viable approach to multi-agent planning. It is worth noting that the idea of using satisfiability for solving the closely related problem of plan merging has been considered in [11]. 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, [11] does not study the parallel encoding of planning into satisfiability that we investigate here.

Another work which extends a classical planner to a multi-agent context is proposed in [12], where they have extended Graphplan [13] for multi-agent planning by proposing Distributed Graphplan. Major difference of their work to ours is that they don't study the positive interactions while coordinating the plans of two agents. Moreover while Distributed Graphplan is only for two agents, μ-SATPLAN finds consistent plans for an arbitrary number of agents. Another major difference is again in the fact that they use the technique of plan merging in which an action can not belong to the joint plan if it is not included in the planning graphs of the individual agents.

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

2 Propositional Satisfiability based Planning:SATPLAN

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 [14]. 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 [15, 16]. Second, it can be easily extended to accommodate the needs of our multi-agent planning scenario.

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 [14]. 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, please refer to [16].

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 $P \models G$.

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 either independent or necessary for the achievement of a common global goal. We restrict ourselves to the

case of two agents, α and β, 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 (cooperative setting) that none of the agents can achieve alone. Moreover, agents have different capabilities. In the most extreme 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 may be either independent or 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 plan 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 may be either independent or 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 (P'_α, P'_β) to the problem such that $max(l(P'_\alpha), l(P'_\beta)) < max(l(P_\alpha), l(P_\beta))$.*

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

To solve the MACAP, we propose a setting in which 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 his plan which does not have any conflict (i.e. *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. We will call such plan of agent β as consistent with the plan of agent α.

The *negative interactions* 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 [17]. 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 $t_1 \leq t \leq t_2$ and A deletes p.

2. *Parallel actions interference*. This conflict was introduced in Graphplan [13]. Two actions interfere if they are executed in parallel and one deletes the preconditions or add effects of the other.

The *positive interactions* allow agents to benefit from the effects of actions performed by other agents in order to achieve their own goals. In this situation agent β could use some effects produced by agent α's actions and therefore avoiding to establish facts that have already been established by agent α.

In the following we will present how agents α and β compute their plans using μ-SATPLAN.

4.1 Independent Plan Computation

Agent α needs to compute his plan and also a set of causal links to provide them to other agents so that no agent threatens them. For this purpose, agent α invokes μ-SATPLAN using 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 returns a plan P_α for the achievement of all the goals or returns $fail$ in the argument P_α. This call also returns the set of causal links C_{P_α} of plan P_α. μ-SATPLAN uses the SATPLAN planner intact to compute plan P_α and calculates the set of causal links C_{P_α} by using the Algorithm 1. This algorithm calculates a set C_P consisting of all the causal links of an input plan P.

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 using 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. To compute such a plan we have proposed following solutions for handling negative and positive interactions.

4.2.1 Handling Negative Interactions

As we have discussed earlier, negative interactions come either from *causal link threatening* or from *parallel actions interference*. Here we explain how to deal with *causal link threatening*.

While constructing his planning graph, agent β checks before putting any operator O at action level i, if it threatens any of the causal links (t_1, t_2, p) from the set C_{P_α} where $t_1 \leq i \leq t_2$. If so then agent β does not put operator O in the planning graph at level i, even though all of its preconditions are satisfied at this level. Agent β

then expands his planning graph by adding new levels in order to reach all his goals and if the problem of threat persists he avoids to use operator *O*. In this situation operator *O* is abandoned.

Example 1 *In order to illustrate, let's consider the well known domain of Blocks World. Let* $ON(X,Y)$ *be a fact meaning that block X is on block Y and* $MOVE(X,Y,Z,i)$ *be an operator meaning that block X moves from Y onto Z at time i. Now let's consider that set* C_{P_α} *has a causal link* $(1,3,ON(a,b))$ *and agent* β *is checking the applicability of the operator* $MOVE(a,b,c,1)$ *in his planning graph. According to our proposal he decides not to put this operator at this level because it threatens a causal link of agent* α*. Moreover he does not put this operator at levels* 2 *or* 3*. Actually* $(1,3,ON(a,b))$ *causal link means that block a should be on block b from time* 1 *to time* 3 *because 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 *then it would spoil the plan of agent* α*.*

We will discuss how to deal with *parallel actions interference* in the following section, where we will also discuss how we deal with *positive interactions*.

4.2.2 Handling Positive Interactions and Parallel Actions Interference

For handling positive interactions, we made the following changes in the classical SATPLAN which are included in μ-SATPLAN.

When agent β starts computing his plan, μ-SATPLAN creates an action for each time step *i* in the plan of agent α. It 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 all those facts added by agent α at time *i*.
- The delete effects of action $NONAME(i)$ are all those facts deleted by agent α at time *i*.
- The preconditions of action $NONAME(i)$ are all those 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 in parallel 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 this, agent β is maintaining the information about the facts added and deleted by agent α at each level.

As it is known the planning graph is then encoded into a CNF format sentence and the solver tries to find a truth value assignment for this sentence. This truth value assignment is actually the solution to the planning problem. Here an important issue arises. The purpose of adding *NONAME* actions in the planning graph of

agent β is to make sure that agent keeps himself up-to-date with the changes made in the environment by agent α, so we also have to ensure that the solver necessarily choose these *NONAME* actions in the solution it finds, otherwise it would not fulfil the purpose. To ensure this, μ-SATPLAN explicitly adds these *NONAME* actions as unary clauses in the CNF sentence. Thus the solver now tries to find a solution including all the *NONAME* actions. This approach has two advantages. The first one is that we deal with *positive interactions* and the second one is that we simultaneously deal with the *parallel actions interference*. In fact as we have added the actions of agent α's plan in the planning graph of agent β, if one action of agent β at level *i* is in interference with an action of agent α at this level, then these actions are automatically declared as mutually exclusive by the planning graph mechanism of μ-SATPLAN. As it is known, if a pair of actions is considered mutually exclusive then both actions can not be executed in parallel. So the solver does not select the action of agent β at level *i*, which interferes with some action of agent α at this level. We illustrate our approach by the following example.

Example 2 *Consider two agents α and β. Agent α has already computed his plan which is $P_\alpha = \{A1(0), A2(0), A3(1)\}$ The positive 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 because there are two time steps in the plan of agent α. The add effects of $NONAME(i)$ are all the effects added by agent α at time i. So in this case $eff(NONAME(0)) = \{eff(A1) \cup eff(A2)\} = \{a0, a1\}$ 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 positive effects namely $prec(B1) = \{a6\}$,$eff(B1) = \{a5\}$, $prec(B2) = \{a5\}$, $eff(B2) = \{a0\}$, $prec(B3) = \{a5\}$, $eff(B3) = \{a7\}$, $prec(B4) = \{a0, a7\}$, $eff(B4) = \{a8\}$. 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. Planning graph of agent β is shown in Figure 1 (Gray lines are NOOPS. Boxes are showing the actions. Small letters followed by numbers are propositions. A line from proposition F to an action O shows that F is the precondition of O. A line from an action O to a proposition F shows that F is add effect of O.) In this figure we can see 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 format sentence. However we have to make sure that the solver necessarily choose the former because otherwise it would mean that effect $a0$ which has already been added by agent α, would be added again by agent β. The solver chooses $NONAME(0)$ because we have explicitly added $NONAME(0)$ and $NONAME(1)$ as unary clause in the CNF sentence and now CNF sentence cannot be made true without assigning a true value to the $NONAME(0)$ and $NONAME(1)$. Agent β's plan will be $P_\beta = \{B1(0), B3(1), B4(2)\}$. It is therefore clear from the plan that agent β does not re-establish the fact $a0$ which has already been established by agent α.*

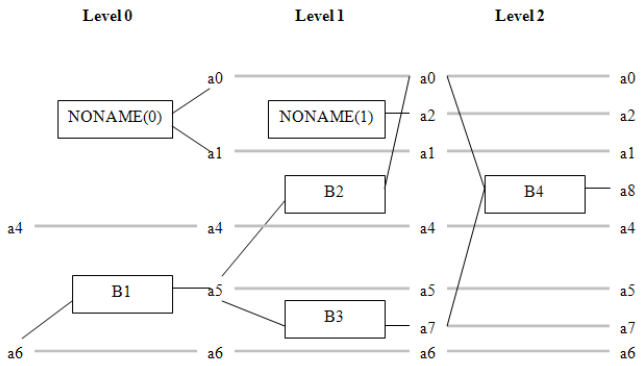

Fig. 1 Illustrating Positive Interactions.

4.3 Coordinated Plan for n^{th} agent

μ-SATPLAN is capable of solving MACAP for n agents where n could be any number greater than one. It means that μ-SATPLAN can be used for finding a plan for the n^{th} 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,..., P_{n-1}. Suppose also that the sets of their causal links are C_1, C_2,..., C_{n-1}. Then a call $ComputeCoordinatedPlan(T_n, G_n, P_1 \cup P_2 \cup .... \cup P_{n-1}, C_1 \cup C_2 \cup .... \cup C_{n-1}, P_n)$ returns a plan P_n for the n^{th} agent which is consistent with the plans of $n-1$ agents.

5 Solving the MAOCAP

To solve the 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, we propose a coordination algorithm (Algorithm 2). Each agent uses the μ-SATPLAN for plan generation, and exchanges messages with the other agent.

First agent α computes his plan P_α using μ-SATPLAN by call *ComputeNewPlan* and sends to agent β as a candidate sub-plan of a joint plan. Then agent β computes a plan P_β consistent with P_α using μ-SATPLAN by call *ComputeCoordinatedPlan* and sends the joint plan (P_α, P_β) to agent α. At this point the joint plan (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 to compute a joint plan. Every time, a joint plan found whose length is less than the current best joint plan, becomes the current best joint plan. This way both agents work out to find an optimal joint plan.

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 moving to the main body of the algorithm, the agents go through a phase in which the algorithm's variables and data structures 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. But, to simplify the presentation, we are not showing it explicitly 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 $max(l(P), l(P'))$, is 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 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 he proposed the 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 but if $P_2 \neq fail$ then agent deletes P_1 from memory because this plan does not lead to an improved joint plan so it does not need to be stored any more. It ensures that at any point there are at most only two plans stored, one which is under consideration and one which is part of the current optimal. Then if possible, a new candidate sub-plan is generated and 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 he does not expect any further candidate sub-plans from other agent. If *continue* variable is true, it generates another plan and sends to other agent, otherwise comes out of 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 $(P_\alpha, fail)$
  if (*continue*) **then**
    Call Sub-Procedure *New_Proposal*
  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 Sub-Procedure *New_Proposal*
  if (*not continue*) and (*not expect*) **then**
    exit(P_{best})
else (ie. $P_\alpha = \emptyset \wedge P = \emptyset$)
  if (*not continue*) **then**
    exit(P_{best})
  if (*continue*) **then**
    expect= false
    Call Sub-Procedure *New_Proposal*
    exit(P_{best})

Sub-Procedure 1 *New_Proposal*

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

6 Experimental Results

In this section we present some preliminary experimental results. An important point to note is that, although the coordination algorithm presented in section 5, is sound and generic to find the global optimal solution to the problem of MACAP, in some cases it could become inefficient. So in our current implementation, in order to re-

No	Problem	Number of Goals A/B	Time for First Plan	Length Pairs of First Plan	Best Length Pairs	Joint Plans Computed	Time for Best Plan
1	AIPSLog11	5/4	3	(9,14)	(13,10)	7	16
2	AIPSLog15	4/7	4	(10,17)	(15,10)	10	28
3	AIPSLog18	4/4	5	(10,14)	(11,11)	3	12
4	AIPSLog20	5/5	7	(14,18)	(14,18)	8	39
5	AIPSLog24	5/4	3	(12,14)	(13,12)	5	14
6	AIPSLog28	5/5	7	(10,16)	(15,13)	7	36
7	AIPSLog30	4/5	4	(12,16)	(12,16)	10	28
8	Storage10	2/2	12	No Plan	(11,18)	16	64
9	Storage12	2/2	32	No Plan	(9,9)	10	216
10	Storage16	3/3	129	No Plan	(13,8)	16	942
11	TPP11	3/3	4	(13,13)	(13,13)	3	9
12	TPP13	4/4	7	(9,14)	(10,11)	5	22
13	TPP14	4/5	8	(7,13)	(9,10)	4	34
14	TPP15	5/5	12	(9,15)	(11,11)	6	50
15	TPP17	6/6	29	(11,11)	(11,11)	3	65
16	TPP19	7/7	47	(10,12)	(11,10)	2	90
17	TPP20	9/6	58	(12,11)	(12,11)	4	117

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

duce the search space, and at the same time to improve the quality of the global solutions found, we do not try to generate every possible proposal plan of an agent. Instead, in our system, the first plan proposed by an agent to another agent is the local optimal generated by SATPLAN and then every subsequent plan has the length which is one more than the length of the previous proposal plan. It means that if the first proposal plan of an agent has length 6 then the next proposal plan of agent will be of length 7. In this way agent generates proposal plans until the length of its proposal plans reaches the current *lbest*. At that moment, it sets the value of *continue* variable to false. So our current implementation of the coordination algorithm does not guarantee the global optimal solution, but comes up with a good solution, which in many cases is much better than the first solution found.

We run μ-SATPLAN and the coordination algorithm on *multi-agent versions* of the well-known planning domain Logistics as well as the Storage and TPP domains from the 5th International Planning Competition [15]. 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. 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 please refer to [15]. 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 refer to the case of two agents. We assume that each agent can execute all actions in the domain, and therefore can achieve its goals without assistance from other agents.

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)

Table 2 A plans pair generated by μ-SATPLAN

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

To understand the efficacy of positive interactions we consider 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 instead he avails the cooperative oppurtunities offered by agent α by using the same truck T1. At time 0, agent β sits idle just waiting for agent α to move T1 to 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 sits 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.

7 Conclusion and future work

In this paper we formailzed the Multi-Agent Coordinated Actions Problem (MACAP) and Multi-Agent Optimal Coordinated Actions Problem (MAOCAP). 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

and MAOCAP. We presented all the details of how μ-SATPLAN deals with negative as well as positive interactions to find consistent plans for multiple agents working in the same environment. Moreover, in this paper we presented, for the first time in multi-agent planning domain, several experimental results that show the added value of adapting SATPLAN for 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.

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.

References

1. Boutilier, G., Brafman, R.: Partial-order planning with concurrent interacting actions. Journal of Artificial Intelligence Research **14** (2001)
2. Katz, M.J., Rosenschein, J.S.: The generation and execution of plans for multiple agents. Computers and Artificial Intelligence **12**(1) (1993) 5–35
3. Ephrati, E., Rosenschein, J.: Divide and conquer in multi-agent planning. In: AAAI94. (1994)
4. Tsamardinos, I., Pollack, M., Horty, J.: Merging plans with quantitative temporal constraints, temporally extented actions and conditional branches. In: AIPS00. (2000)
5. Tonino, H., Bos, A., Weerdt, M.D., Witteveen, C.: Plan coordination by revision in collective agent-based systems. Artificial Intelligence **142,2** (2002) 121–145
6. Cox, J., Durfee, E.: Discovering and exploiting synergy between hierarchical planning agents. In: AAMAS03. (2003)
7. Cox, J., Durfee, E.: An efficient algorithm for multiagent plan coordination. In: AAMAS05. (2005)
8. Steenhuisen, J., Witteveen, C., Mors, A., Valk, J.: Framework and complexity results for coordinating non-cooperative planning agents. Lecture Notes in Computer Science **4196** (2006) 98
9. Ottens, B., Faltings, B.: Coordinating agent plans through distributed constraint optimization. In: 18th International Conference On Automated Planning And Scheduling, ICAPS. (2008)
10. Dimopoulos, Y., Moraitis, P.: Multi-agent coordination and cooperation through classical planning. In: Proceedings of the IEEE/WIC/ACM Intern. Conf. on Intelligent Agent Technology (IAT). (2006) 398–402
11. A.Mali: Plan merging and plan reuse as satisfiability. In: 5th European Conference on Planning, ECP99. (1999)
12. Iwen, M., Mali, A.: Distributed graphplan. In: Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI2002). (2002) 138–145
13. Blum, A., Furst, M.: Fast planning through planning graph analysis. Artificial Intelligence Journal **90**(1-2) (1997)
14. Kautz, H., McAllester, D., Selman, B.: Encoding plans in propositional logic. In: Principles of Knowledge Representation and Reasoning, KR. (1996)
15. Gerevini, A.: 5th international planning competition: Results of the deterministic truck. Available from http://zeus.ing.unibs.it/ipc-5/
16. Kautz, H., Selman, B., Hoffmann, J.: SATPLAN: Planning as satisfiability. In: Booklet of the 2006 International Planning Competition. (2006) Available from http://zeus.ing.unibs.it/ipc-5/publication.html.
17. Weld, D.: An introduction to least commitment planning. AI Magazine **15**(4) (1994)