Multi-Agent Planning as a Dynamic Search for Social Consensus

Eithan Ephrati
Jeffrey S. Rosenschein*
Computer Science Department
Hebrew University
Givat Ram, Jerusalem, Israel

Abstract

When autonomous agents attempt to coordinate action, it is often necessary that they reach some kind of consensus. Reaching consensus has traditionally been dealt with in the Distributed Artificial Intelligence literature via negotiation. Another alternative is to have agents use a voting mechanism; each agent expresses its preferences, and a group choice mechanism is used to select the result. Some choice mechanisms are better than others, and ideally we would like one that cannot be manipulated by untruthful agents.

Coordination of actions by a group of agents corresponds to a group planning process. We here introduce a new multi-agent planning technique, that makes use of a dynamic, iterative search procedure. Through a process of group constraint aggregation, agents incrementally construct a plan that brings the group to a state maximizing social welfare. At each step, agents vote about the *next* joint action in the group plan (i.e., what the next transition state will be in the emerging plan). Using this technique agents need not fully reveal their preferences, and the set of alternative final states need not be generated in advance of a vote. With a minor variation, the entire procedure can be made resistant to untruthful agents.

1 Introduction

The field of Distributed Artificial Intelligence (DAI) is concerned with coordinated, efficient activity by groups of autonomous agents. Activity in multi-agent worlds often requires agreement by the agents as to how they will act, and the reaching of consensus is a major concern of DAI. The formation of *multi-agent plans* has been approached in several different ways: through the use of synchronization techniques [Georgeff, 1984], such as those used in operating systems, through the distribution of single-agent planners [Corkill, 1979], such as NOAH, and through centralized planners that ensure coordination [Rosenschein, 1982].

In this paper, we present a new approach to deriving multi-agent plans. We consider how agents could reach consensus using a voting procedure, without having to reveal full goals and preferences (unless that is actually necessary for consensus to be reached). Our technique also does away with the need to generate final alternatives ahead of time (instead, candidate states arise at each step as a natural consequence of the emerging plan). The agents iteratively converge to a *plan* that brings the group to a state maximizing social welfare.

2 The Scenario

Our scenario involves a group of n agents operating in a world currently in the state s_0. Each agent has its own private goal. Since all agents operate in the same environment and may share resources, it is desirable that they agree on a joint plan for the group that will transform the world to an agreed-upon final state.

Each agent is assumed to have a value, a *worth*, that it associates with states that satisfy its goal. This worth can be used to assign a utility to any state, goal or not, that is reached by a particular plan (the techniques for doing this assignment are discussed below).

Given the existence of such a worth function, we want to give the agents a method for choosing a group plan. The agents will all participate in the choice process, as well as in carrying out the resulting multi-agent plan. The group plan should then result in a compromise state that is in consensus.[1]

2.1 An Example

Consider a simple scenario in the slotted blocks world. There are four slots (a, b, c, d), five blocks $(1, 2, 3, 4, 5)$, and the world is described by the relations: $\mathbf{On}(Obj_1, Obj_2)$—$Obj_1$ is stacked onto Obj_2; $\mathbf{Clear}(Obj)$—there is no object on Obj; and $\mathbf{At}(Obj, Slot)$—Obj is located at $Slot$. The function $\mathbf{loc}(Obj)$ returns the location (slot) of Obj. Slots themselves function as (stationary) objects (e.g., block 1 in slot b could be described by $On(1, b)$).

*This research was partially supported by the Israeli Ministry of Science and Technology (Grant 032-8284)

[1] The decision procedure, to be presented below, may also serve in a single-agent planning scenario, where an agent is trying to integrate a group of disparate goals.

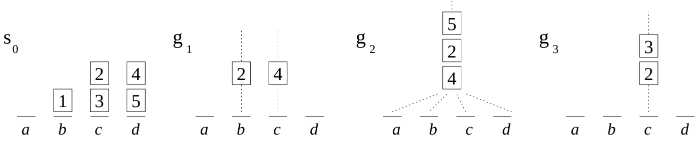

Figure 1: A Blocks World Example

There are three agents operating in the world. The start state is shown at the far left of Figure 1. As further represented in that figure, these agents have (respectively) the following goals: $g_1 = \{At(4,c), At(2,b)\}$, $g_2 = \{On(2,4), On(5,2)\}$, $g_3 = \{On(3,2), At(2,c)\}$, with respectively the the individual worths: 8, 12 and 16.

There is only one available operator: **Move**(Obj_1, Obj_2)—place Obj_1 onto Obj_2. This operator can be characterized by the following STRIPS-like lists:
[$Prec$: $Clear(Obj_1)$, $Clear(Obj_2)$, $On(Obj_1, Obj_x)$],
[Del: $On(Obj_1, Obj_x)$, $Clear(Obj_2)$, $At(Obj_1, loc(Obj_1))$],
[Add: $On(Obj_1, Obj_2)$, $At(Obj_1, loc(Obj_2))$)].
Assume that when a single agent performs the $Move$ operation there is a cost of 4, while if two agents $Move$ an object together the operation costs a total of 3 (1.5 each).

We want the agents to jointly choose a plan that results in a compromise state.

3 General Overview of the Process

The algorithm enables a group of agents to find the state that maximizes their social welfare function f^U. There are many possible global utility functions, such as taking the product of individual agent utilities, or taking their median, or taking their sum. Our procedure here is not dependent on any particular function, and is equally suitable regardless of the one chosen. Given the set of individual utility functions of the agents in the decision group, we define the Social Welfare/Utility $U(E)$ (of any set of constraints E) to be that function f^U of all the individual worth functions, minus the cost of achieving the set from s_0 (that is, $U(E)$ equals f^U minus the cost of achieving E).

The underlying idea is the *dynamic generation of alternatives* that locates the most desirable state for the society. At each step, all agents reveal additional information about their private goals. The current set of candidate states is then expanded and (possibly) pruned to comprise the new set of candidate states. The process continues until all newly formed sets of constraints have lower social utility than their ancestor sets. Note that our primary concern here is not with the complexity of the planning process but rather with the resulting state of the multi-agent plan.

The search procedure needs an accurate value for the social utility of each candidate state in order to proceed correctly (i.e., the search space of alternatives is dynamically pruned by the social welfare criterion). To provide this value, the agents vote over the set of candidates at each step.

The search method combines aspects of hill-climbing with breadth-first search. For example, parallel searches are carried out in each promising direction (where social utility is growing or constant within the gap bound ∇_{f^U} [2]). When the search encounters a direction where social utility decreases beyond this limit, however, the search is terminated (reminiscent of hill-climbing).

The procedure has the following advantages: **(a)** alternatives are generated by the entire group dynamically (allowing the procedure to be distributed [Ephrati and Rosenschein, 1993]); **(b)** utilities will be calculated and submitted only for "feasible" alternatives (utilities of infeasible alternatives need not be revealed, reducing the choice procedure's computational complexity, and also respecting agent privacy when possible [Ephrati and Rosenschein, 1992b]); **(c)** agents are required to submit only the minimally "conflict-sufficient" information about their goals (described below), further maintaining their privacy.

3.1 Definitions

- $e(g_i)$ is the set of absolutely necessary constraints needed for any optimal plan to achieve the goal g_i, starting at the initial state s_0. In accordance with the partial order over these constraints,[3] we divide $e(g)$ into subsets of constraints. Each such subset within $e(g)$ comprises all the constraints that can be satisfied within j (optimal) steps, and are necessary at some subsequent step after j. The total number of operators (the length of the plan) that satisfies the set of constraints $e(g)$ is denoted by $l(e(g))$.

  We denote $e(g)$'s components by $\bigcup_j E_g^j$, such that E^j includes all the constraints that can be satisfied within j steps, and are necessary at some step $\geq j$. For any $j \geq l(e(g))$, we define E^j to be the description of the goal g.

  Example: In the blocks world scenario from Section 2.1, $e(g_1)$ for Agent 1 would be: $E_1 = [C(2), C(4)](= E_1^1) \cup [C(4), A(2,b)](= E_1^2) \cup [A(4,c), At(2,b)](= E_1^3)$

- A constraint $I \in E^j$ is said to be *temporary* if later in the plan there is a constraint $\bar{I} \in E^k$ (where $k > j$) that denies it ($I \wedge \bar{I} \models F_{alse}$). We say that a set of apparently conflicting constraints E is semi-consistent ($E \not\models_{tmp} F_{alse}$) if the removal of temporary constraints makes it consistent (see Section 4.1 for an example). Given a semi-consistent E we define $r(E)$ to be the set of all maximal consistent subsets of the predicates in E.

- $P(E)$ denotes the set of the cheapest "grounded" (or "complete" [Chapman, 1987]) plans that achieve the final subset of E. We denote these states by $s(E)$ ($= \{s \mid (s \models E^{l(E)}) \wedge c(s_0 \rightsquigarrow s) = \min_{k \models E^{l(E)}} c(s_0 \rightsquigarrow k)\}$).

[2] Given f^U and the agents' utility functions, we define the gap bound ∇_{f^U} to be the maximal gap between any local maximum of f^U and any local minimum that follows it.

[3] Constraints are temporally (partially) ordered sets of the domain's predicates associated with the appropriate limitations on their codesignation.

- $F_{ollow}(E)$ is defined to be the set of constraints that can be satisfied by invoking at most one operator, given the initial set of constraints E ($F_{ollow}(E) = \{I \mid \exists op \exists P[op(P(E)) \models I]\}$).

4 The Algorithm

This section describes the algorithm in more detail, along with a running example. At each step of the procedure, agents try to impose more of their private constraints on the group's aggregated set of sets of constraints. Since agents want to maximize their own utility, they will impose as many constraints as they can at each step. The set of all non-pruned aggregated sets of constraints at step k is denoted by $\mathcal{A}^k$ (its constituent sets will be denoted by A_j^k, where j is simply an index over those sets). $\mathcal{A}^{k+}$ denotes the set $\mathcal{A}^k$ before it is pruned (similarly, its non-pruned components are denoted by A_j^{k+}).

As an example, assume a simple scenario of the slotted blocks world (we use the same operators and predicates as described in Section 2.1). There are 3 blocks (1,2,3) and two agents (a_1, a_2). The initial state is $\{On(1,b), On(3,1), On(2,c)\}$. The agent's goals are (respectively) $g_1 = \{On(1,2)\}$ and $g_2 = \{On(2,3)\}$.[4]

The exact procedure is defined as follows:

1. At step 0 each agent i finds $e(g_i)$—the individual temporally ordered set of constraints that achieves the goal g_i, starting from the initial state. The virtual set of alternatives is initialized to be the empty set ($\mathcal{A}^0 = \emptyset$).

   In our example, we have:[5] $e(g_1) = \{[\mathtt{C}(2)] \cup [\mathtt{C}(2), C(1)] \cup [O(1,2)]\}$ (this ordered set induces the plan $\langle M(3,\bar{2}), M(1,2)\rangle$) and $e(g_2) = \{[\mathtt{C}(3), C(2)] \cup [O(2,3)]\}$ (inducing the plan $\langle M(2,3)\rangle$). Note that $\mathtt{C}(2)$ and $\mathtt{C}(3)$ are temporary since they are denied later in the plan.

2. At step k, each agent may declare $E_i^{A_j^k} \subseteq E_i^{l+1}$ only if any $k \leq l$ E_i^k was already declared and accepted by the group, and the declaration is "feasible," i.e., it can be reached by invoking one operator on some set of constraints that was reached by the procedure during the previous step: $\exists A_j^k[(A_j^k \in \mathcal{A}^k) \wedge (\bigcup_{n=1}^{l} E_i^l \subseteq A_j^k) \wedge (E_i^{A_j^k} \subseteq F_{ollow}(A_j^k))]$. i can try to impose elements of his "next" private subset of constraints on the group decision only if they are still relevant and his previous constraints were accepted by the group.

   At the first step, each agent i may declare $E_{g_i}^1$. In our example, this will be $E_{g_1}^1 = [\mathtt{C}(2)]$ and $E_{g_2}^1 = [\mathtt{C}(3), C(2)]$.

   At the second step, a_1 declares $E_1^{A_1^1}$, which in this example equals $E_{g_1}^2 = [\mathtt{C}(2), C(1)]$. Similarly, a_2 declares $E_{g_2}^2 = [O(2,3)]$. (Both are in $F_{ollow}(\mathcal{A}^1)$, which contains only one subset.) At the third step, a_1 declares $[O(1,2)]$ and a_2 declares $[O(2,3)]$ (his final goal, which is already in $\mathcal{A}^2$).

3. For each set of constraints $A_j^k \in \mathcal{A}^k$, we generate all the maximal consistent or semi-consistent extensions $\{A_j^{(k+1)+}\}$ with elements of $\bigcup_i E_i^{A_j^{(k+1)+}}$ (i.e., each element in $A_j^{(k+1)+}$ is defined as $\{A_j^k \cup \{I \mid (I \in \bigcup_i E_i^{A_j^{(k+1)+}}) \wedge ((I \cup A_j^{(k+1)+}) \not\models_{tmp} F_{alse})\}\}$).

   At the first step, the aggregated set of constraints is $\{\mathtt{C}(3), C(2)\}$. At the second step, both declarations may coexist consistently, and there is therefore only one successor to the previous set of constraints: $\{\mathtt{C}(2), C(1), O(2,3)\}$. At the third step, the aggregated set is $\{O(1,2), O(2,3)\}$, and it satisfied both agents' goals.

4. At this stage, all extensions are evaluated so as to enable the pruning of sets that reduce social utility. Each agent declares the utility it associates with each newly-formed set of aggregated constraints.

   The first set of aggregated constraints is satisfied by the initial state, and thus induces the null plan (with cost of zero). The value given to that first set is the value that agents assign to the initial state. The second set can be achieved by the plan $\langle M(3,a)\rangle$, with cost of 3. In order for the set not to be pruned, it must be the case that the agents value the set by at least $3 + \nabla_{f^U}$ more than the initial set. The third set is the set that satisfies the social welfare criterion (by our assumption above), and therefore (by definition) will have higher value than previous steps (and not be pruned).

5. The next set of sets of constraints is pruned so that it contains only sets that do not decrease the social utility (with respect to their ancestor set) by more than the gap bound ∇_{f^U}. Formally, $\mathcal{A}^{k+1} = \{A_j^{k+1} \mid A_j^{k+1} \in \mathcal{A}_j^{(k+1)+} \wedge \forall l \leq k[U(A_j^{k+1}) \geq U(A_j^l) + \nabla_{f^U}]\}$.

   In this simple example, all the individual extensions are mutually consistent. Therefore, there is only one aggregated extension at each step. Since this set brings the agent closer to the set that satisfies the social welfare criterion, it is not pruned.

6. The process ends when $\mathcal{A}^{k+1} = \mathcal{A}^k$. The values assigned by each agent to each state in $\mathcal{A}^k$ are then compared to find the consistent sets that maximize social utility ($\hat{\mathcal{A}}^k = \{A \mid A \in \mathcal{A}^k \wedge \forall A^*[A^* \in \mathcal{A}^k \Rightarrow U(A^*) \leq U(A)]\}$). Among these equivalent sets, one is randomly chosen.

   In the example there is only one such set—the final set which is determined in the fourth step.

Theorem 1 *Given any social welfare function f^U, and an appropriate gap bound ∇_{f^U}, this mechanism finds all sets that maximize the Social Welfare. Consensus will be reached after at most $O(\max_A l(P(A))$ steps such that $P(A)$ results in a state that maximizes the group's social welfare.*

Proof. The proof of this theorem, and the one below, appear in [Ephrati and Rosenschein, 1992a] with slightly different notations. □

[4] This is reminiscent of Sussman's Anomaly in the single-agent planning scenario—where the plan to achieve one subgoal obstructs the plan that achieves the other. The example assumes that although the final state that satisfies both agents' goals costs 9 to reach, it is the state that meets the social welfare criterion.

[5] We will use the first letter to denote the full operator predicate, and $\bar{y}$ to denote any location excluding y's. We also use a typewriter font to denote temporary constraints.

4.1 Constraints

There are several important aspects of, and requirements for, the procedure above to succeed, which we discuss in this section. Figure 2 shows three simple scenarios in the slotted blocks world that will help us explain the issues involved. In these examples, two agents can achieve a consensus state that fully satisfies both agents' goals. There are three slots, and several blocks in each world. Only the *Move* operator is available, and it costs 2 under all circumstances.

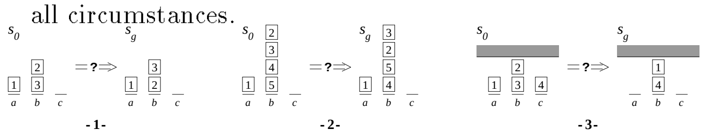

Figure 2: Three scenarios in the slotted blocks world

4.1.1 Specification of Constraints

An important requirement for the success of the procedure is that each agent identify and declare only the absolutely necessary constraints needed for its individual plan to succeed. As an example, consider the first scenario in Figure 2. Two agents want to achieve the following goals: $g_1 = \{A(1,a), C(1)\}$ and $g_2 = \{O(3,2), A(2,b)\}$. These two goals can co-exist, as shown in the final goal state. To achieve his goal, a_1 need not take any action. a_2, on the other hand, has to carry out $\langle M(2,x), M(3,\bar{2}), M(2,b), M(3,b)\rangle$. The only way for the second step of this plan to be completed is by stacking either block 3 or block 2 onto block 1. By including either of these into his set of constraints, a_2 would encounter a_1's opposition, thus (perhaps) preventing his own goal from being achieved.

However, the purpose of the second *Move* operator is just to achieve $A(3,\bar{b})$. If a_2 declares his set of constraints to be $[C(2)] \cup [C(2), C(3)] \cup [C(2), C(3), A(3,\bar{b})] \cup [C(2), C(3), A(2,b)] \cup [A(2,b), O(3,2)]$, the conflict is avoided.

4.1.2 Aggregation of Temporary Constraints

In the second scenario, a_1's goal is the same, but a_2's goal is now $\{O(5,4), O(3,2), A(2,b), A(4,b)\}$. As in the previous example, there is no way for him to achieve his goal without trying to temporarily violate a_1's goal. But in contrast to that example, there is no way to specify a_2's constraints in a way that would avoid the conflict. Here, for example, $A(2,\bar{b})$, although temporary, is crucial for a_2's plan's success. Were the process to consider sets of constraints to be relevant only as long as they were consistent, the process might stop after two steps, avoiding the goal state from even being considered.

It is therefore necessary that the *temporary* conflicts between the agents' plans not cause deadlocks (that is, if there is a temporal order that can later resolve them). This phenomena is achieved by allowing the existence of semi-consistent sets of constraints, and having agents express preferences over these sets. For that reason, agents should recognize what their temporary constraints are (terms that will be violated by their own future actions). Identifying the temporary constraints of his own plan, a_2's set of constraints (E_2) would then become: $[\mathtt{C}(2)] \cup [\mathtt{C}(2), \mathtt{A}(2,\bar{\mathtt{b}})] \cup [\mathtt{A}(2,\bar{\mathtt{b}}), \mathtt{C}(4), \mathtt{A}(3,\bar{2}\bar{\mathtt{b}})] \cup [\mathtt{A}(2,\bar{\mathtt{b}}), \mathtt{C}(4), \mathtt{A}(3,\bar{2}\bar{\mathtt{b}}), \mathtt{C}(5), \mathtt{A}(4,\bar{\mathtt{b}})] \cup [\mathtt{A}(2,\bar{\mathtt{b}}), \mathtt{C}(4), \mathtt{A}(3,\bar{2}\bar{\mathtt{b}}), \mathtt{C}(5), \mathtt{A}(5,\bar{4}\bar{\mathtt{b}})] \cup [\mathtt{A}(2,\bar{\mathtt{b}}), \mathtt{C}(4), \mathtt{A}(3,\bar{2}\bar{\mathtt{b}}), \mathtt{C}(5), \mathtt{A}(5,\bar{4}\bar{\mathtt{b}}), \mathtt{C}(2), A(4,b)] \cup [\mathtt{A}(3,\bar{2}\bar{\mathtt{b}}), C(2), A(4,b), O(5,4), C(3)] \cup [C(2), A(4,b), O(5,4), C(3), A(2,b)] \cup [A(4,b), O(5,4), O(3,2), A(2,b)]$
As can be seen in this specification, all the constraints that contradict a_1's goal are temporary. Therefore, even though a_2's plan actually violates a_1's goal, the mutual goal state is reachable.

4.1.3 Aggregation of Functional Constraints

The generation of consensus sets of constraints is based on the aggregation of the individual sets of constraints. As the third example in Figure 2 shows, this is not always trivial. Here, there is a ceiling that makes it impossible for more than two blocks to be stacked. This time a_1's goal is $A(1,b)$ while $g_2 = A(4,b)$. We assume that the function $h(b)$ returns the height (in number of blocks) at slot b. As in all our scenarios, s_0 satisfies E_i^1 of both agents.[6] Following the second step of his plan, each of the agents has as temporary constraints $\mathtt{h(b)} \leq \mathtt{1}$ and $C(x)$, where x is the block he wants to move onto slot b. These constraints enable each agent to move "his" block to slot b after removing block 2.

The aggregation of these constraints requires careful analysis. First, it must be recognized when the aggregated constraints of two identical terms such as $h(x) \leq 1$ will be $h(x) \leq 2$, or $h(x) \leq 1$, or even $h(x) \leq 0$. Had block 4 been located on block 1 in the initial state of our example, the third solution would be appropriate. Using it in the given scenario, however, would yield as the aggregated set of constraints in the second step $[C(1), C(4), h(b) \leq 0]$. The induced states of this set cost 10 move operators, while the actual plan that achieves the mutual goal costs only 5. The problem here is that both constraints are *temporary* (each agent needs a free space for one block only momentarily). Thus, in this case, the aggregated height should stay 1, leading to a state that is only three *Move* steps distant from the initial state. Unfortunately, it not clear how in general this subtle analysis is to be done.

4.1.4 Evaluation of Sets of Constraints

Evaluation of sets of constraints plays an important role in the search procedure. One straightforward worth function for an arbitrary set A might be built by taking the worth of a goal state (assumed to be available), subtracting the cost of the single-agent plan from A to the goal, then subtracting the agent's share of the cost of the multi-agent plan to get from start state s_0 to A.

Note, however, that using the above equation the worth of s_0 for an agent would simply be the worth of his goal, minus the cost of his one-agent plan to reach that goal (in a one-agent scenario this would be true for every set). Thus, since the evaluation function does not capture the notion of progress in the plan, the agent has no motivation to carry out his plan at all.

There are several ways to refine the worth function so as to solve this problem. One way is by making the

[6] $E_1^1 = [C(1), C(2), \mathtt{h(c)} \geq 1]$ and $E_2^1 = [C(4), C(2), \mathtt{h(a)} \geq 1]$. Note that by its nature, the height constraint is temporary, since it always is a precondition of an action that violates it!

"future-cost" (i.e., $w_i(g_i) - c_i(A \rightsquigarrow g_i)$) more sensitive to the progress of the plan. A simple approach is to take into consideration only that fraction of the goal's worth which reflects the amount of work already done to achieve it ($\approx w_i(g_i) \times [l(P(s_0 \rightsquigarrow A))/l(P(A \rightsquigarrow g_i))]$, which is meaningful only if $w_i(g_i) > c_i(s_0 \rightsquigarrow g_i)$). Another way is to give greater weight to the cost of operators that are located further along in the plan ($\approx w_i(g_i) - \sum_1^k k \times C(op_k)$). Or, assuming that each operator has a probability ($pr(op_k)$) associated with its success, we could use $\approx (\prod_1^k pr(op_k) \times w_i(g_i)) - c_i(A \rightsquigarrow g_i)$. These evaluations may be further refined by having weighted costs and/or probability of success associated with each of the constraints that needs to be achieved in order to transform the given set into the goal set (see [Haddawy and Hanks, 1990] and [Kanazawa and Dean, 1989] for richer probabilistic approaches).

Note that instead of assigning worth to sets of constraints, it may sometimes be more natural to evaluate their induced states ($s(A)$ instead of A). In any case, the worth associated with all states induced from a single set of constraints will be equivalent. In addition, note that for many variations of the above worth functions, it will be sufficient to take the gap bound ∇_{f^U} to be zero (what was called a progressive worth function in [Ephrati and Rosenschein, 1992a]). For example, it would be sufficient to assume above that $\forall i c(A_1 \rightsquigarrow A_2) \leq c_i(s_1 \rightsquigarrow s_2)$).

5 Manipulative/Insincere Agents

In choosing a state that maximizes social welfare, it is critical that agents, at each step, express their true worth values. However, if our group consists of autonomous, self-motivated agents, each concerned with its own utility (and not the group's welfare), they might be tempted to express false worth values, in an attempt to manipulate the group choice procedure. This is a classic problem in voting theory: the expression of worth values at each step can be seen as an (iterative) cardinal voting procedure, and we are interested in a non-manipulable voting scheme so that the agents will be kept honest. We have investigated other aspects of this problem in previous work [Ephrati and Rosenschein, 1992b].

Fortunately, there do exist solutions to this problem, such that the above plan choice mechanism can be used even when the agents are not necessarily benevolent and honest. If the social welfare function is taken to be the (weighted) sum f^U_{sum} (or average) of the individual utilities, it is possible to ensure that all agents will vote honestly. This is done by minor changes to the procedure of Section 4, that allow it to use a variant of the Clarke Tax mechanism (CTm).

In [Ephrati and Rosenschein, 1991] we proposed the CTm as a plausible group decision procedure. The basic idea of the mechanism is to make sure that each voter has only one dominant strategy, telling the truth. This phenomenon is established by choosing the alternative that scores the highest sum of bids/votes and then taxing some agents. The tax (if any) equals the portion of the agent's bid for the winning alternative that made a difference to the outcome. Given this scheme, revealing true preferences is the dominant strategy.

In our procedure, the agents are participating in many intermediate votes, and alternatives are generated dynamically (as a consequence of the intermediate votes). Therefore, the original version of the CTm cannot be used efficiently. Instead we use an iterative variation of the CTm; at each step, the tax is defined with respect to all previous steps, but is actually levied only at the final step. We use the following definitions for the stepwise Clarke tax mechanism:

- The function $v_i : \mathcal{A} \rightarrow \mathbb{R}$, returns the *true* worth (to a_i) of each aggregated set A. Similarly, the function $d_i^k(j)$ returns the *declared* worth of the set A_j by agent a_i at step k. $\vec{d_i^k}$ denotes the vector $\langle d_i^k(1), \ldots, d_i^k(m)\rangle$, the agent's declared worth over all alternatives and $\vec{v_i^k}$ denotes the true value.
- The profile of preferences declared by all agents at step k is denoted by D_n^k, where $D_{\neg i}^k$ denotes this set excluding i's preferences, such that $D_n^k = (D_{\neg i}^k, \vec{d_i^k})$.
- The choice function $f : D_n^k \times \mathcal{A} \rightarrow \mathcal{A}$ returns the state that is the *maximizer* of $\sum_{i=1}^n d_i^k(A)$;
- The tax imposed on i at step k is $t_i^k(f(D_n^k)) = \sum_{j \neq i} d_j^k(f(D_{\neg i}^k)) - \sum_{j \neq i} d_j^k(f(D_{\neg i}^k, \vec{d_i^k}))$, if this value is positive. Otherwise, t_i^k will be zero. Therefore, the utility $u_i^k(f(D_n^k))$ of agent i with respect to the chosen alternative is $w_i(f(D_n^k)) - t_i^k(f(D_n^k))$.

The planning algorithm itself should also be updated in two ways. First, since each intermediate vote is only over a subset of candidates, there is the possibility that an agent will "shift" his vote by a constant, keeping a single round's preferences accurate while undermining inter-vote comparisons. To maintain truth telling as a dominant strategy, it is necessary that artificial shifting does not occur. Therefore, we will require that all votes be relative to some "benchmark": we include A^0 (the empty set) in the set of alternatives at every step. If each agent is motivated to give his true preferences over the other states relative to A^0 ($v_i(A)$), then the score of each state s in the vote is exactly $f^U_{sum}(A)$.

Second, the tax is calculated with respect to the final choice. Knowing that a *semi-consistent* set cannot be chosen, an agent might give it an artificial value in order to change the final outcome. We therefore allow agents to vote only over consistent sets. Step 4 of the algorithm is therefore changed as follows:

- For each $A \in \mathcal{A}^{(k+1)+} \setminus \mathcal{A}^k$ find $r(A)$, its maximal consistent subsets. Each agent gives its vote regarding each state in $r(A) \cup A^0$. The worth of a consistent set is simply the sum of individual worths given to that state (note that for any consistent set A, $r(A) = A$). The worth of each semi-consistent set in $\mathcal{A}^{(k+1)+}$ is computed as follows: for each agent and semi-consistent set A, there is a consistent set with maximal worth in $r(A)$. The worth of a semi-consistent set is taken to be the sum of these maximal worth sets in $r(A)$, over all agents ($\sum_i \max_{E \in r(A)} w_i(E)$).

At the and of the process, each agent is fined the Clarke Tax with respect to the final group choice.

Theorem 2 *At* any *step k of the procedure, i's best strategy is to vote over the alternatives at that step ($\mathcal{A}^k$) according to his* true *preferences $\vec{v_i}$.*

5.1 Using the Procedure on Our Example

We now use the iterative procedure with its CTm to solve the first problem presented in Section 2.1. We assume that the cost of reaching a state is divided equally among the agents (by side-payments if necessary), and that each agent i uses the worth function $w_i(A) = w_i(g_i) - c_i(s(A) \rightsquigarrow g_i)$ (the goal's worth minus the cost of the work needed to transform a state induced by the set to the goal state). From the agents' individual goals, we get the following constraints:
$E_1 = [C(2), C(4)](= E_1^1) \cup [C(4), A(2,b)](= E_1^2) \cup [A(4,c), At(2,b)](= E_1^3)$[7] $E_2 = [\mathtt{C}(2), \mathtt{C}(4)] \cup [\mathtt{C}(4), \mathtt{C}(2), C(5)] \cup [C(2), C(5), O(2,4)] \cup [O(2,4), O(5,2)]$ $E_3 = [\mathtt{C}(2)] \cup [\mathtt{C}(2), C(3)] \cup [C(2), C(3), \mathtt{A}(3,\bar{\mathtt{c}})] \cup [\mathtt{C}(2), C(3), A(2,c)] \cup [A(2,c), O(3,2)]$

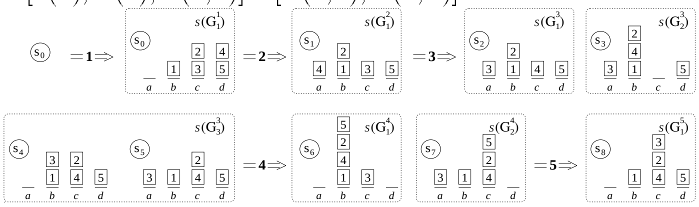

Figure 3: Induced States of the Five Steps

Figure 3 presents the induced states at each step (in this example, all the generated sets are consistent). At the first step, each agent declares E_i^1. $\mathcal{A}^1$, the set that includes all possible consensus sets of constraints, has only one member: $A_1^1 = \{[C(2), C(4)]\}$. $s(A_1^1)$ also has only one member, s_0. Agents then vote on this state, and it receives a score of 0 (for example, a_2 goal's worth is 12, and to achieve it the agent would perform $\langle M(4,\bar{c}), M(2,4), M(5,2)\rangle$ at a cost of 12; therefore, he values s_0 as 0).

At the second step, each agent hands in E_i^2 (which is in $F_{ollow}(A_1^1)$ for each i). Since all these constraints coexist consistently, $\mathcal{A}^2 = [C(2), C(4), C(5), C(3), A(2,b)]$. This set induces the single state s_1 ($= s(A_1^2)$) as described in Figure 3. Note that there are many other states that could satisfy this set of constraints, but s_1 has the *minimal cost*. This state can be achieved by $\langle Move(4,a), Move(2,1)\rangle$; therefore, the state costs 6. Subtracting this cost from the worth values given by each agent (4 in this case by all three agents), the state scores 6. Since this score is greater than that of the preceding state s_0, the process continues.

[7] a_1's plan might be $\langle Move(2,1), Move(4,3)\rangle$. The first operation is enabled since the constraint $C(2)$ is satisfied. Th $C(2)$ constraint is satisfied by s_0, which is included in E_1^1. $C(4)$ is needed for a future operator, but it is also satisfied by s_0, and therefore it too is included in E_1^1. $A(2,b)$ can be satisfied within one move, and is necessary at all future times for the plan to succeed, so it is included in any future set of constraints.

At the third step, the new added constraints generate 3 possible maximally consistent extensions: $A_1^3 = [A(2,b), A(4,c), C(2), C(5), C(3), A(3,\bar{c})]$ inducing s_2, $A_2^3 = [O(2,4), A(2,b), C(2), C(5), C(3), A(3,\bar{c})]$ inducing s_3, and $A_3^3 = [O(2,4), A(4,c), C(2), C(5), C(3), A(3,\bar{c})]$ inducing s_4 and s_5. These induced states respectively score 11, 3, 12, and 12; A_2^3, which decreases the social utility, is therefore pruned.

At the fourth step, the two remaining extensions are extended further; a_1 hands in $\{A(2,b), A(4,c)\}$ (which is in $F_{ollow}(A_1^3)$), a_2 hands in $\{O(2,4), O(5,2)\}$ and a_3 $\{C(3), C(2), A(2,c)\}$ (both in $F_{ollow}(A_3^3)$). These constraints yield six different extensions that again induce the states s_2, s_3, s_4, s_5 and the new states s_6 and s_7 (that score respectively -5 and 5). Therefore, only the sets that induce s_4 and s_5 can be further extended (by a_3) to induce s_8. Although s_8 fully satisfies a_3's goal, it scores only 5 and the process ends.

All intermediate votes are now gathered for the final vote. Both s_4 and s_5 maximize the social welfare utility (both are one operation distant from each of the agents' goals). Both a_2 and a_3 are taxed 2. a_2 improves its utility by 2 and a_3 by 6 (a_1's utility is not improved with respect to s_0). The group's social utility is therefore improved by 8.

6 Related Work

There are a number of artificial intelligence researchers whose work relates to the approach we have been discussing above. Some of this work follows in the footsteps of Korf [Korf, 1987], who showed that the planning search space can be reduced if the final goal can be decomposed into several sub-goals, and the plans that achieve these sub-goals can be combined to achieve the original goal. This result suggests that the multi-agent planning algorithm presented in this paper can also serve to reduce the search space in a single-agent planning scenario if a non-optimal solution is acceptable.

A similar approach is taken in [Nau *et al.*, 1990] to find an optimal plan. It is shown there how planning for multiple goals can be done by first generating several plans for each subgoal and then merging these plans. Finding the solution is guaranteed (under several restrictions) only if a sufficient number of alternative plans is generated for each sub-goal. Our approach does away with the need for several plans for each subgoal by using constraints instead of grounded plans (a level of abstraction that represents all possible grounded plans).

In [Foulser *et al.*, 1992] it is shown how to merge *grounded linear plans* (as opposed to aggregating constraints) in a dynamic fashion. To achieve an optimal final plan it takes that algorithm $O(\prod_{i=1}^{n} l(P(g_i)))$, while the approximation algorithm that is presented there takes polynomial time.

Our approach also resembles the GEMPLAN system [Lansky, 1990]. There, the search space of the global plan is divided into "regions" of activity. Planning in each region is done separately, but an important part of the planning process within a region is the updating of its overlapping regions (in our terms, all individual plans are generated and aggregated simultaneously). This model served as a basis for the DCONSA system [Pope *et al.*, 1992] where agents were not assumed to have complete information about their local environments. The combination of local plans was done through "interaction constraints" that were pre-specified.

The concept of solution that our algorithm employs (maximization of social welfare) also resembles the approach taken in CONSENSUS [Clark *et al.*, 1992] where several expert systems "elect" a plan that scores the highest rating with respect to the individual points of view. There, however, the election refers to different complete global plans that each expert generates.

Another advantage of our proposed process is that it can easily be modified to deal with dynamic priorities. Since the search is guided by the vote taken at each step, it is possible to allow the agents to change their "tastes" or priorities over time (for example, due to environmental changes). As an example, in the Multi-Fireboss Phoenix system [Moehlman and Lesser, 1990] planning (the actions needed to assess and contain fires) is performed by several spatially distributed agents. The system addresses, through a sophisticated negotiation protocol, the dynamic allocation of resources. Our algorithm would solve this problem in a direct manner, without negotiation. At each time interval, the agents would vote over the possible relevant distributions (one step of the algorithm per time interval). Given the individual utilities, the accurate distribution of resources would be chosen that maximizes the social utility (minimizes the damage according to the group's perspective). In addition (as mentioned in Section 5), there is no need to assume that the agents are benevolent.

7 Conclusions

We have introduced a new dynamic, iterative voting procedure. It enables a group of agents to construct a joint plan that results in a final state that maximizes social welfare for the group. The technique is more direct and formally specified than other consensus procedures that have been proposed, and maintains agent privacy more effectively. Techniques such as these provide a natural method for the coordination of multi-agent activity. Conflicts among agents are then not "negotiated" away, but are rather incrementally dealt with. Agents iteratively search for a final state that maximizes the entire group's utility, incrementally constructing a plan to achieve that state. The search can be constructed so that any manipulation by an untruthful agent will harm the agent more than it helps him.

References

[Chapman, 1987] D. Chapman. Planning for conjuctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[Clark *et al.*, 1992] R. Clark, C. Grossner, and T. Radhakrishnan. ONSENSUS: a planning protocol for cooperating expert systems. In *Proceedings of the Eleventh International Workshop on Distributed Artificial Intelligence*, pages 77–94, Glen Arbor, Michigan, February 1992.

[Corkill, 1979] D. Corkill. Hierarchical planning in a distributed environment. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 168–175, Tokyo, August 1979.

[Ephrati and Rosenschein, 1991] E. Ephrati and J. S. Rosenschein. The Clarke Tax as a consensus mechanism among automated agents. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 173–178, Anaheim, California, July 1991.

[Ephrati and Rosenschein, 1992a] E. Ephrati and J. S. Rosenschein. Multi-agent planning as search for a consensus that maximizes social welfare. In *Pre-Proceedings of the Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, Rome, Italy, July 1992.

[Ephrati and Rosenschein, 1992b] E. Ephrati and J. S. Rosenschein. Reaching agreement through partial revelation of preferences. In *Proceedings of the Tenth European Conference on Artificial Intelligence*, pages 229–233, Vienna, Austria, August 1992.

[Ephrati and Rosenschein, 1993] E. Ephrati and J. S. Rosenschein. Distributed consensus mechanisms for self-interested heterogeneous agents. In *First International Conference on Intelligent and Cooperative Information Systems*, Rotterdam, May 1993. To appear.

[Foulser *et al.*, 1992] D. E. Foulser, M. Li, and Q. Yang. Theory and algorithms for plan merging. *Artificial Intelligence*, 57:143–181, 1992.

[Georgeff, 1984] M. Georgeff. A theory of action for multi-agent planning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 121–125, Austin, Texas, August 1984.

[Haddawy and Hanks, 1990] P. Haddawy and S. Hanks. Issues in descision-theoretic planning: Symbolic goals and numeric utilities. Technical report, University of Illinois at Urbana-Champaign, IL, 1990.

[Kanazawa and Dean, 1989] K. Kanazawa and T. Dean. A model for projection and action. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

[Korf, 1987] R. E. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88, 1987.

[Lansky, 1990] A. L. Lansky. Localized search for controlling automated reasoning. In *Proceedings of the Workshop on Innovative Approachess to Planning, Scheduling and Control*, pages 115–125, San Diego, California, November 1990.

[Moehlman and Lesser, 1990] T. Moehlman and V. Lesser. Cooperative planning and decentralized negotiation in Multi-Fireboss Phoenix. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 144–159, San Diego, November 1990.

[Nau *et al.*, 1990] D. S. Nau, Q. Yang, and J. Hendler. Optimization of multiple-goal plans with limited interaction. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 160–165, San Diego, California, November 1990.

[Pope *et al.*, 1992] R. P. Pope, S. E. Conry, and R. A. Mayer. Distributing the planning process in a dynamic environment. In *Proceedings of the Eleventh International Workshop on Distributed Artificial Intelligence*, pages 317–331, Glen Arbor, Michigan, February 1992.

[Rosenschein, 1982] J. S. Rosenschein. Synchronization of multi-agent plans. In *Proceedings of the National Conference on Artificial Intelligence*, pages 115–119, Pittsburgh, Pennsylvania, August 1982.