A Heuristic Technique for Multiagent Planning

Eithan Ephrati
Computer Science Department
Bar Ilan University
Ramat Gan, Israel
tantush@bimacs.cs.biu.ac.il

Jeffrey S. Rosenschein
Institute of Computer Science
The Hebrew University
Jerusalem, Israel
jeff@cs.huji.ac.il

Abstract

The subject of multi-agent planning has been of continuing concern in Distributed Artificial Intelligence (DAI). In this paper, we suggest an approach to multi-agent planning that contains heuristic elements. Our method makes use of subgoals, and derived sub-plans, to construct a global plan. Agents solve their individual sub-plans, which are then merged into a global plan. The suggested approach reduces overall planning time and derives a plan that approximates the optimal global plan that would have been derived by a central planner, given those original subgoals.

We explore three different scenarios. The first involves a group of agents with a common goal. The second considers how agents can interleave planning and execution when planning towards a common, though dynamic, goal. The third examines the case where agents, each with their own goal, can plan together to reach a state in consensus for the group.

Finally, we consider how these approaches can be adapted to handle rational, manipulative agents.

1 Introduction

1.1 Multi-Agent Planning

The term *planning*, in classical Artificial Intelligence (AI), usually refers to the generation of sequences of actions leading from an initial state to a goal state. Most planning research in AI has historically been concerned with a single agent, but more recently the term *multi-agent planning* has come to be used for the generation of coordinated sequences of actions by or for groups of agents.

The field of Distributed Artificial Intelligence (DAI) is concerned with coordinated, efficient activity by groups of autonomous agents. Given this characterization of the field, it is clear that one of the central missions of DAI is to explore multi-agent planning. The term "planning," however, can be used in many different ways, and DAI researchers have used the term as a catch-all phrase that characterizes very different streams of work.

For example, one focus of DAI planning research has been that of "planning for multiple agents," which considers issues inherent in centrally directed multi-agent execution. Smith's Contract Net [45, 46] falls into this category, as does other DAI work such as [43, 39, 26]. A second focus for research has been "distributed planning," where multiple agents all participate in coordinating and deciding upon their actions [5, 44, 10, 12, 41].

The question of whether the group activity is fashioned centrally or in a distributed manner is only one axis of comparison. Another important issue that distinguishes DAI research in multi-agent planning is whether the goals themselves need to be adjusted, that is, whether there may be any fundamental conflicts among different agents' goals. Thus, for example, Georgeff's early work on multi-agent planning assumed that there was no basic conflict among agent goals, and that coordination was all that was necessary to guarantee success [20, 21, 47]. Similarly, planning in the context of Lesser, Corkill, and Durfee's research often involves coordination of activities (e.g., sensor network computations) among agents who have no inherent conflict with one another (though surface conflict may exist). "Planning" here is multi-agent coordination of actions that helps avoid redundant or distracting activity, causes efficient exploration of the search space, etc.

Another important issue is the relationship that agents have to one another, e.g., the degree to which they are willing to compromise their goals for one another (assuming that such compromise is necessary). *Benevolent Agents* [44] are those that, by design, are willing to accommodate one another; they have been built to be cooperative, to share information, and to coordinate in pursuit of some (at least implicit) notion of global utility. These benevolent agents for the most part characterize research in the DAI subfield of Cooperative Problem Solving. In contrast, research in the subfield of Multi-Agent Systems posits the existence of self-motivated agents, perhaps built by different designers, with purely local notions of utility. Each will cooperate only when it is in its best interest to do so [57]. Still another potential relationship among agents is a modified master-slave relationship, called a "supervisor-supervised" relationship, where non-absolute control is exerted by one agent over another [16].

The synthesis, synchronization, or adjustment process for multiple agent plans thus constitute some of the (varied) foci of DAI planning research. Synchronization through conflict avoidance [20, 21, 47], distribution of a single-agent planner among multiple agents [4], and the use of a centralized multi-agent planner [43], have all been explored.

1.2 Concerns of this Work

In this paper, we present a heuristic approach to deriving multi-agent plans (earlier versions of this work have been partially reported on in [17, 11, 15]). By making use of the computational power of multiple agents working in parallel, the process is able to reduce the total elapsed time for planning as compared to a central planner. After providing a general overview of the technique (Section 2), we consider three possible planning scenarios in multi-agent environments.

In the first scenario (Section 3), a group of agents has to cooperatively achieve one common global goal. We suggest an approach to multi-agent planning that contains heuristic elements. Our method makes use of subgoals, and derived sub-plans, to construct a global plan. Agents solve their individual sub-plans, which are then merged into a global plan. The suggested approach reduces overall planning time and derives a plan that approximates the optimal global plan (i.e., the multi-agent plan of lowest cost) that would have been derived, given those original subgoals.

In the second scenario (Section 4), the agents again need to cooperatively achieve one common global goal, but they are now operating under time constraints. Because of these time constraints, they are unable to compute a globally optimal course of action; instead, local decisions are made and execution is interleaved with planning. We also assume that the global common goal may dynamically change during the process.

In the third scenario (Section 5), each agent has its own private, individual goal. We consider how agents could reach consensus about what multi-agent plan to carry out 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. Our approach makes use of the aggregation of constraints, and in Section 6 we elaborate on this issue.

In Section 7 we discard an implicit assumption that was made in the previous sections, namely the assumption that agents will benevolently follow the rules of the procedures. We show how the procedures can be adjusted to deal with self-interested, rational agents. Finally, we conclude with a brief comparison to related work in Section 8.

2 General Overview of the Technique

The approach described in this paper allows a group of agents to derive a global coherent plan given their individual goals and (associated) plans. The main contribution of this work is a formal technique that can guide the construction of a global multi-agent plan in a variety of multi-agent scenarios while handling the possible interactions among individual plans.

2.1 An Example – One Common Global Goal

Consider a scenario in the discrete grid sticky blocks world.[1] There are ten blocks ($a, \ldots, j$) in the world. All blocks have a 1 square foot base; six have a length of 2 feet and four a length

[1]We say "discrete grid" because the ground is marked with locations that specify where the blocks can go. The blocks are sticky and can support one another horizontally, as is clear in Figure 1. While the blocks world is inappropriate for studying many real-world issues in robotics, it is more suitable for the study of abstract goal interactions.

of 3 feet. The ground surface is divided into 30 contiguous marked regions (coordinates) of 1 square foot each, aligned in a row.

The world may be described by the following relations:

Clr(b)—there is no object on b;

On$(b, x, V/H)$—b is located on block x (or at location x) either vertically (V) or horizontally (H);

At(x, loc)—the left edge of object x (agent or block) is at loc.

The functions $gr(b)$ and $l(b)$ return the grid region of b's left edge, and the length of b, respectively.

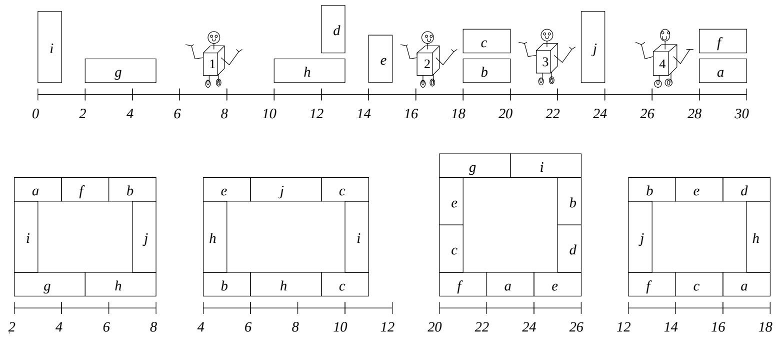

Figure 1: An Arch in the Sticky Blocks World Example

There are four agents in the world. The start state is shown at the top of Figure 1. The goal is to build a (symmetric) window of inner size $\geq (3 \times 4)$ square feet. The bottom of Figure 1 shows several possible instantiated goal states.

The available operators (described in a STRIPS-like fashion) are:[2]

Take$_i(b, x, y)$—take b from x to y, where x and y denote either a region or another block.

[**cost:** $|loc(x) - loc(y)| \times l(b)$,

prec: $On(b, x, z), Clr(b), Clr(y), At(a_i, x), At(b, x)$

del: $Clr(y), At(a_i, x), At(b, x)$,

add: $On(b, y, z), At(a_i, y), At(b, y)$]

Rot$_i(b)$—rotate b by $\pm\frac{\pi}{2}$.

[**cost:** $l^2(b)$,

prec: $Clr(b), At(b, y), At(a_i, y), On(b, x, z)$,

del: $On(b, x, z)$,

add: $On(b, x, \bar{z})$] (where $\bar{H} = V$ and vice versa)

Move$_i(x, y)$—Go from x to y.

[**cost:** $|x - y|$, **prec:** $At(a_i, x)$,

del: $At(a_i, x)$,

[2] We assume in this specific example that all agents have identical capabilities. The operators are indexed with respect to the agent that performs them. When the identity of the performing agent is unimportant, we sometimes omit the subscript.

add: $At(a_i, y)]$

The final goal may be divided into subgoals in several ways. For example, the global goal might be described by specifying four corners, or (alternatively) by specifying the construction of a base, two columns, and a top (we will be using the latter subgoal division). Given such an *a priori* breakdown into subgoals, our agents are to go through a planning process that will result in satisfying the original global goal. Note that in the case where agents have individual goals to begin with, the final *global* goal is not known. In such a scenario, the global goal is identified through a search process.

2.2 The High-Level Algorithm

At the beginning of the process, each agent generates a sufficient set of alternative sub-plans that derive its individual subgoal from the initial state, while ignoring the other agents' sub-plans. These sub-plans are then "merged," incrementaly, to form one coherent multi-agent plan that achieves the global goal.

The search for the global plan is through the space of states. A partial plan (i.e., a "path") in the queue of alternative plans is identified by the states that it induces. Given some specific initial state, the agents will go through the following loop until they derive a sufficiently good plan to reach a desirable final state. The notions of sufficiently good plan and desirable final state will be clarified in the following sections.

Until A sufficiently good global plan is found do:

1. Delete the first partial plan from the queue and identify all its plausible successor states. Each successor state represents a feasible set of constraints,[3] that will be satisfied given that path. The sets of constraints that are relevant for further exploration are determined by aggregating the sets of constraints that each agent declares, based on its individual plan.
2. Determine the additional sequence of multi-agent steps needed to achieve each successor state. These additional steps are concatenated onto the current, incomplete, multi-agent plan.
3. Based on the individual plans and the actual path that leads to each successor, determine the heuristic value of each newly formed path. The heuristic functions in different scenarios below correspond to variants of the A^* algorithm.
4. Merge each of the new successors into the queue according to its heuristic value.

[3]Constraints are temporally ordered sets of the domain's predicates associated with the appropriate limitations on their codesignation [2]. Using SNLP [34] terminology, these are the propositions in the causal links that construct the "nonlinear abstraction" of p_i, partially ordered according to their safety conditions, and stated as prerequisites (preconditions in UCPOP's terminology [40]). For example, if step w has prerequisite $On(x, B)$ and step s enables it by establishing $On(A, B)$, the essential proposition is $On(x, B)$ rather than $On(A, B)$. The term "constraint" is used because these preconditions act as constraints on the search for a global plan.

Essentially, the process exploits the possible decomposition of a global goal into subgoals.

2.3 Decomposition Reduces Complexity

The complexity of the planning process is measured by the time (and space) consumed. Let b be the branching factor of the planning problem (the average number of new states that can be generated from a given state by applying a single operator), and let d denote the depth of the problem (the shortest path from the initial state to the goal state). The time complexity of the planning problem is then $O(b^d)$ [27].

In a multi-agent environment, where each agent is capable of carrying out each of the possible operators (possibly with differing costs), the complexity may be even worse. A centralized planner would need to consider assigning each operator to each agent. Thus, finding an optimal plan becomes $O(n \times b)^d$ (where n is the number of agents). Furthermore, if the possible parallelism in the execution time is to be taken into account, all possible simultaneous combinations should be considered.

However, if the global goal can be decomposed into n subgoals ($\{g_1, \ldots, g_n\}$), the time complexity may be reduced significantly. Let b_i and d_i denote respectively the branching factor and depth of the optimal plan that achieves g_i (in general $b_i \approx \frac{b}{n}$ and $d_i \approx \frac{d}{n}$). Then, as shown by Korf [27], if the subgoals are independent or serializable,[4] the central multi-agent planning time complexity can be reduced to $\sum_i((n \times b_i)^{d_i})$.[5]

This phenomenon of reduced complexity due to parallel search can be exploited most naturally in a multi-agent environment. The underlying idea is to assign to each agent a subgoal and let that agent construct the plan that achieves it. Since agents plan in parallel, planning time is further reduced to $\max_i(n \times b_i)^{d_i}$. Moreover, if each agent generates its plan independently (assuming that the available operators are common knowledge) then the complexity becomes $\max_i(b_i)^{d_i}$. The global plan is then constructed out of local plans that are based upon the agents' local knowledge.

Unfortunately, unless the subgoals are independent or serializable, planning using this subgoal technique does not ensure finding the optimal *global* plan. Furthermore, the plans that achieve the set of subgoals usually interfere, and conflicts (or redundant actions) may arise.

Our approach to multi-agent planning exploits this same phenomenon of parallel search, while also containing heuristic elements. Our method makes use of pre-existing subgoals, and derived sub-plans, to construct a global plan. Subgoals are not necessarily independent, nor are they necessarily serializable. The essential idea is that the individual sub-plans serve to derive a heuristic function that is used to prune the global search space. The separate

[4]A set of subgoals is said to be *independent* if the plans that achieve them do not interact. If the subgoals are *serializable* then there exists an ordering among them such that achieving any subgoal in the series does not violate any of its preceding subgoals [27].

[5]The summation is over n subplans. In the search for the multiagent plan, the branching factor of each subplan is $n \times b_i$ instead of simply b_i, since the planner needs to consider the assignment of each operator to each agent.

agents' sub-plans, each derived separately and in parallel, are ultimately merged into a unified, valid global plan. The suggested approach reduces overall planning time and derives a plan that approximates the optimal global plan that would have been derived, given those original subgoals. This approach also removes the need for a central planner that has global knowledge of the domain and of the agents involved.

3 The Scenario of Planning by Subgoals

This scenario involves a group $A = \{a_1, \ldots, a_n\}$ of n agents. These agents are to achieve a global goal G. The global goal, G, has been divided into m subgoals ($\{G_1, \ldots, G_m\}$), and formulated as a subgoal planning problem (i.e., the interrelationship and order among subgoals has been specified). The agents communicate as they construct a global plan. In the remainder of this paper we will assume, without loss of generality, that $m = n$.[6]

3.1 Assumptions and Definitions

- The *global goal*, G, is a set of predicates, possibly including uninstantiated variables. Each instance g of G is simply a grounded instance of G (a set of grounded predicates that specifies a set of states). We assume that G is divided into n abstract subgoals $[G_1, G_2, \ldots, G_n]$, such that there exists a consistent set of instances of these subgoals that satisfies G ($\cup_i g_i \models G$).

  Example: The subgoals described in Section 2.1 may be described as a sequence of adjusted horizontal blocks $\{b_1, \ldots, b_k\}$ such that $gr(b_i) = gr(b_{i-1}) + l(b_{i-1})$ for all $i > 1$. Thus, the base/top subgoal may be informally specified as: $\{On(b_1, x, H), On(b_2, x + l(b_1), H), \ldots, On(b_k, gr(b_{k-1}) + l(b_{k-1}), H) \mid \sum_k l(b_k) \geq 6\}$. Similarly, a column can be described as $\{On(b_1, x, V), On(b_2, b_1, V), \ldots, On(b_k, b_{k-1}, V) \mid \sum_k l(b_k) \geq 3$.

- In general there exists some interdependency among subgoals, such as the required (partial) order of their achievement and some of their physical attributes (i.e., location, size, weight etc.). We denote the set of constraints (and domain axioms) that specifies the desired interface between subgoals by $D(G)$.

  We assume that for a given set of instances of subgoals that models G and satisfies $D(G)$, there exists some multi-agent plan that achieves G by achieving the subgoals.

  Example: For the subgoals described in Section 2.1 the interface constraints, $D(G)$, may be described as follows: $\{l(col_1) = l(col_2), l(base) = l(top),\ gr(base) = gr(col_1) = gr(top), gr(col_2) = gr(col_1) + l(base) - 1,\ (Blocks(col_1) \cap Blocks(col_2) \cap Blocks(base) \cap Blocks(top)) = \emptyset\}$. The intuitive meaning of $Blocks$ is the set of blocks making up the

[6] If $m > n$, we will have more than one goal assigned to some agents, and if $n > m$, we will have some agents uninvolved in the planning process.

structural component; we constrain different parts to use different blocks (i.e., structural components are always distinct and never share a block).

- Any plan that achieves G_i is denoted by p_i. Each such sub-plan is a sequence of operators $\langle op_1, \ldots, op_m\rangle$ that ends in some instance, g_i, of G_i, starting at some instance, g_i^0, of G_i^0 (the preconditions for op_1, i.e., the initial state for the subgoal G_i).

  Usually, G_i^0 will simply be the initial state, for most i. However, if there exists some partial order over the achievement of subgoals, G_i^0 may specify some combination of other subgoals and/or intermediate states of the plans that achieve them.

- Each agent has a *cost function* ($C: OP \times \mathcal{A} \Rightarrow \mathbb{R}$) over the domain's operators [18]. The cost of a_j's plan $c_j(p_j)$ is defined to be $\sum_{k=1}^{m} c_j(op_k)$. Other, more general definitions of a cost function are possible [11], but are beyond the scope of this paper.

3.2 General Overview of the Merging Process

In this section, we will consider the primary phase of our subgoal technique, namely merging the sub-plans that achieve the given subgoals. A sub-plan is constructed by an agent with only a local view of the overall problem. Therefore, conflicts may exist among agents' sub-plans, and redundant actions may also have been generated. Given the set of sub-plans, we are looking for a method to inexpensively merge them into an optimal global plan.

As mentioned above, constraining the plan to satisfy fixed subgoals may derive a non-optimal plan. Consider again the scenario presented in Section 2.1. There, we have specified subgoals that describe straight elements of the window (top, side, etc.). Therefore, any pre-existing corner (such as the one created by blocks d and h in the start state) will be either ignored or destroyed; an optimal global plan might have used it. The quality of the solution returned by our process is limited by this inherent sub-optimality of restricting ourselves to fixed subgoals.

The entire process is illustrated in Figure 2. We employ a cost-driven merging process that results in a coherent global plan, given the sub-plans. As shown in Part **(a)** of Figure 2, we assume that each agent, i, has been assigned (only for the purposes of the *planning process*) one instance of a subgoal g_i. Given that instance, the agent will derive p_i, the sub-plan that achieves it.

As illustrated in Part **(b)** of Figure 2, the process is iterative. The underlying idea is the *dynamic generation of alternatives* that identifies the optimal global plan. At each step, all agents state additional information about the sub-plan of which they are in charge. The current set of candidate global plans is then expanded to comprise the new set of candidate global plans. The process continues until a sufficiently good plan is found (as described below). Plans are represented as sets of absolutely necessary sets of constraints that enable them (the gray arrows in the figure).

As illustrated in Part **(c)** of Figure 2, the search method essentially employs an A^* algorithm. The heuristic function ($f' = g + h'$) that guides the search is dynamically determined

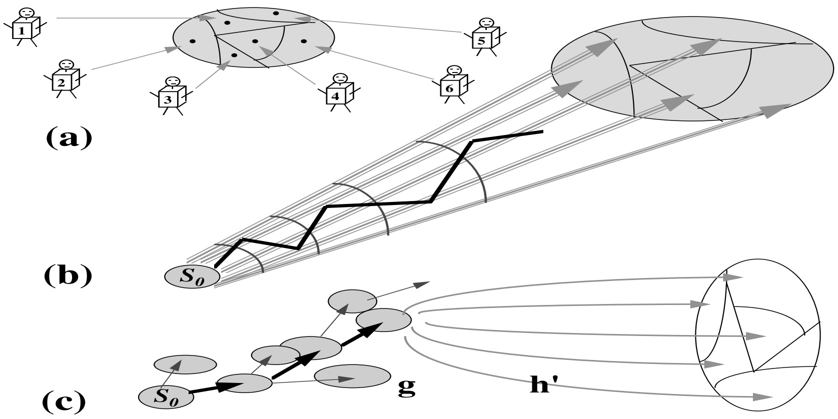

Figure 2: Overview of the Process

by the agents during the process. g is the cost of the partial multi-agent plan that has already been constructed. h' is the sum of the approximate remaining costs, h'_i, that each agent assigns to his own sub-plan. Since it is based upon an actually constructed plan, each individual estimate, h'_i, is accurate, ignoring sub-goal interaction. Thus, if the subgoals are independent, the global heuristic function ($\sum_i^n h'_i$) will be accurate, and the merging process will choose the correct (optimal) candidate for further expansion at each step of the process.

Unfortunately, since sub-plans will tend to interfere with one another, in general, h' is an underestimate (the individual estimates will turn out to be too optimistic). An underestimate heuristic evaluation is also desirable, since it will make the entire A^* search admissible, meaning that once a path to the global goal has been found, it is guaranteed to be the optimal one. However, due to overlapping constraints ("positive" interactions) the global heuristic evaluation might sometimes be an overestimate. Therefore, the A^* search for the optimal path must continue until the potential effect of misjudgment in the global heuristic evaluation fades away. In general, the more overlap that appears in the individual sub-plans, the more additional search steps are needed.

The procedure has the following advantages: **(a)** alternatives are generated by the entire group dynamically (allowing the procedure to be distributed [14]); **(b)** conflicts and "positive" interactions are addressed within a unified framework; **(c)** As mentioned above in Section 2, the complexity of the merging process is significantly reduced. One can see in Part **b** of Figure 2 a kind of "cone" that represents the fact (pointed out above) that the branching factor of the search space is strictly constrained by the individual plans' constraints.

3.3 More Definitions

Having presented the broad outlines of the merging process, we now introduce some additional necessary definitions.

- $\hat{e}_i(g_i)$ is the set of absolutely necessary general constraints[7] needed for some plan to achieve the grounded instance of the subgoal g_i, starting with the initial state g_i^0. It is thus a set of sets of constraints. Each element of the top-level set corresponds to some nonlinear plan that would have been generated by any classical POCL planner. We assume that each agent has access to such a planner. In accordance with the partial order over these constraints, we divide $\hat{e}_i(g_i)$ into subsets of constraints. Each such subset within $\hat{e}_i(g_i)$ 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 these subsets is denoted by $\mid \hat{e}_i(g_i) \mid$.

- For the merging/composition process to find the (optimal) global plan, it will, in general, be necessary to generate more than just one plan for some subgoals. d_i denotes the depth (radius) of the search that is needed so as to generate the sufficient number of sub-plans that achieve g_i. We assume that d_i is known ahead of time (for each g_i).[8]

- A refinement of $\hat{e}_i(g)$ is either a different temporal order of that set that would still achieve g, or a further specification (extended partition) of its sub-sets. $\mathcal{E}_i(g)$ denotes the (sufficient) set of plans that is generated within this d_i-depth search with their refinements.

  We denote the components of $e(g_i)$ (any member of $\mathcal{E}(g_i)$) by $\bigcup_j E_i^j$, such that E_i^j includes all the conditions that can be satisfied within j steps, and are necessary at some step $\geq j$. For any $j \geq \mid e(g_j) \mid$, we define E^j to be the description of the goal g_i.
  Example: Assume in the sticky blocks world scenario from Section 2.1, that g_1 is determined to be $On(g,h,H), At(g,11), At(h,11)$. Then $\hat{e}_1(g_1)$ would be: $\{[Clr(g), Clr(d)](= \hat{E}_1^1)\cup\ [At(a_j,gr(d)), Clr(g),\ Clr(d)](=\ \hat{E}_1^2)\cup\ [Clr(g), Clr(h)](=\ \hat{E}_1^3)\cup\ [At(a_i,gr(h)), Clr(g), Clr(h)]\cup\ [At(h,11)),\ On(h,11,H), Clr(g), Clr(h)]\cup[At(h,11)),\ On(h,11,H), Clr(g), Clr(h), At(a_i,gr(g))]\cup[At(h,11)), On(h,11,H),\ At(g,11), On(g,h,H)]\}$ (inducing the plan $\langle Move(gr(a_i), gr(d)), Take(d,12,9)\ , Move(9,10), Take(h,10,11), Move(11,2), Take(g,2,11)\rangle$).

  One refinement of this plan would replace $\hat{E}_1^2$ with $[At(a_j,gr(g)), Clr(g), Clr(d)]\cup[At(g,9), Clr(g), Clr(d)]\cup[At(a_j,gr(d)), Clr(g), Clr(d)]$ (inducing the plan $\langle Move_1(gr(a_i), gr(g)),$

[7] Recall that we use the term constraints in the same way as it is used by Chapman [2], i.e., in the sense of preconditions of a STRIPS-like planner.

[8] This powerful assumption is needed only for the *completeness* of the global planning process. However, using domain dependent knowledge, the corresponding d_i's may be assessed heuristically. In general, the more the sub-plans will tend to interact (and the closer to optimal the solution needs to be) the deeper the d_i's that are needed.

$Take_1(g, 2, 7)$, $Move_1(gr(a_i), gr(d))$, $Take_1(d, 12, 14)$, $Move_1(14, 10)$, $Take_1(h, 10, 11)$, $Move_1(11, 7)$, $Take_1(g, 7, 11)\rangle$).

- $P(s_0 \leadsto E)$ (or just $P(E)$ for short) denotes the set of "grounded" plans (what Chapman calls "complete" plans [2]) that is induced by the set of constraints E. The length of each such plan is denoted by $l(P(s_0 \leadsto E))$.

- $p_1 \bowtie p_2$ denotes the concatenation of the plans p_1 and p_2. Operators that are invoked simultaneously in a multi-agent plan are grouped together. The order within this group implies the order in which they are completed (e.g., $\langle Op_1, Op2, \ldots \{Op_j, Op_k\}, \ldots, Op_m\rangle$ denotes the fact that Op_j and Op_k are initiated simultaneously and Op_j will be terminated prior to Op_k).

- $F^1_{\text{ollow}}(E)$ 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 ($F^1_{\text{ollow}}(E) = \{I \mid \exists op \exists P[P(E) \bowtie op \models I]\}$). Similarly $F^2_{\text{ollow}}(E)$ is the set of constraints that can be satisfied by invoking at most two operators *simultaneously* (by two agents) given E, and $F^n_{\text{ollow}}(E)$ is the set that can be achieved by at most n simultaneous actions.

- An *interface point* $I(p, \{p_j\})$ between a plan, p, and a set of plans, $\{p_j\}$, is a constraint (a proposition or an equality/inequality) that is needed for the success of p, but may contradict (under some bindings or ordering) some constraint that is expressed in one of the plans in $\{p_j\}$ (similar to a "negative threat" in classical POCL planners). The interface set $\mathcal{I}(p, \{p_j\})$ is the set of interface points. In the example, each binding of a block (or any other resource) is a possible interface point.

  An interface point $I \in E$ is said to be *soft* if changing the binding (codesignation) of a variable that it specifies may cause the same change to all other appearances of that variable throughout the plans $P(E)$, but may not change the structure of these plans. In the example, the location of each element of the window (actually the location of the window itself) is a soft interface point.

3.4 The Multi-Agent Planning Algorithm for a Common Goal

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.

The set of all aggregated sets of constraints at step k is denoted by $\mathcal{A}^k$ (its constituent sets will be denoted by A^k_j, where j is simply an index over those sets). $\mathcal{A}^{k+}$ denotes the set that has the maximal value according to the heuristic function at step k.

As an example, consider a simpler scenario of the sticky blocks world, as described in Figure 3 (we use the same operators and predicates as described in Section 2.1). There are three agents (a_1, a_2, a_3) and 4 blocks (a,b,c,d) with lengths of $1, 2, 2,$ and 3 respectively. The initial state is $\{At(a_1, 2)$, $At(a_2, 4)$, $At(a_3, 6)$,$At(a, 2)$, $On(a, c, H)$,$At(b, 4)$,$On(b, d, V)At(c, 1)$,

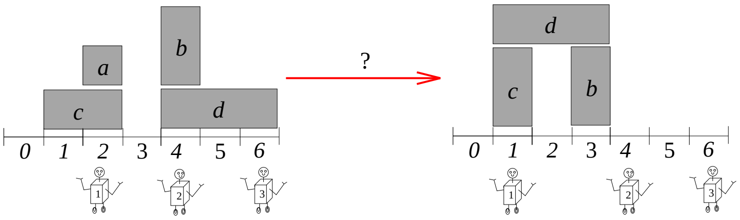

Figure 3: A Small Arch in the Sticky Blocks World Example

$On(c, 1, H)$,$At(d, 4)$,$On(d, 4, H)\}$. The agents are to construct an arch made of $\{\{LeftColumn, RightColumn\}, Top\}$. The agent's subgoals are (respectively):

$g_1 = \{At(c, 1), On(c, 1, V), Clr(c)\}$,
$g_2 = \{At(b, 3), On(b, 3, V), Clr(b)\}$, and
$g_3 = \{At(d, 1), On(d, c, H)\}$.

To simplify things we will use throughout this example only $\hat{e}_i$ instead of $\mathcal{E}_i$, and $F^1_{\text{ollow}}(A^{k+})$ instead of $F^n_{\text{ollow}}(A^{k+})$. The resulting multi-agent plan is illustrated in Figure 4.

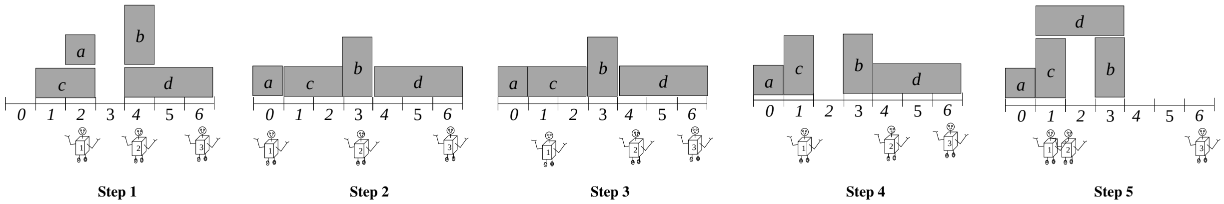

Figure 4: The resulting multi-agent plan

The exact procedure is defined as follows:

1. At step 0 each agent i finds $\mathcal{E}_i(g_i)$—the set of all alternative absolutely necessary temporally ordered sets of constraints that achieve the subgoal g_i, starting from some initial configuration g_i^0, and is consistent with $D(G)$ and its refinements. (Actually, it is more efficient to have each agent find $\hat{e}_i(g_i)$ at the beginning of the process and generate its refinements, as necessary, as the process proceeds.)

   The set of alternatives is initialized to be the empty set ($\mathcal{A}^0 = \emptyset$).

   In our example, we have:[9] $\hat{e}_1(g_1) = \{[Clr(a), \underline{At(c, 1)}, At(a_i, gr(a))]^{[0]} \cup [\underline{Clr(c)}]^{[1]} \cup [At(a_i, gr(c))]^{[2]} \cup [On(c, 1, V)]^{[4]}\}$ (this ordered set induces the plan $\langle Take_1(a, 2, 3)$, $Move_1(3, 1)$, $Rot_1(c)\rangle$).

   $\hat{e}_2(g_2) = \{[\underline{Clr(b)}, At(a_j, gr(b)), Clr(3)]^{[0]} \cup [\underline{On(b, 3, V)}]^{[2]}\}$ (inducing the plan $\langle Take_2(b, 4, 3)\rangle$).

[9]We underline constraints that, once satisfied, must stay valid throughout the process. The region $\bar{b}$ denotes any region besides $gr(b)$. The additional estimated cost of satisfying a subset of constraints appears in the superscript brackets.

And $\hat{e}_3(g_3) = \{[Clr(b), Clr(3)]^{[0]} \cup [At(a_k, gr(b)), Clr(3)]^{[2]} \cup [\underline{Clr(d)}]^{[2]} \cup [At(a_k, gr(d)]^{[1]} \cup [\underline{\hat{e}_2(g_2)}, \underline{\hat{e}_1(g_1)}, \underline{On(d,c,H)}]^{[9]}\}$ (inducing the plan $\langle Move_3(6,4), Take_3(b,4,3), Move_3(3,4), Take_3(d,4,1)\rangle$).

Notice that there exists a positive relation between a_2's and a_3's sets of constraints (both would need block b to be removed from on top of block d), but there is a possible conflict between their plans and a_1's set of constraints (all count on the vacancy of slot 3).

2. At step k one aggregated set (of constraints), A_j^{k+}, is chosen from all sets with minimal heuristic value, $\mathcal{A}^{k+}$. This set (with its corresponding multi-agent plan) is the path currently being considered. Each agent declares $E_i^* \subseteq E_i^{l+1}$ only if for all $k \leq l$ E_i^k is already modeled by A_j^{k+}. That is, each agent declares the maximal set of propositions, E_i^*, such that: (**a**) These constraints represent some possible sequence of consecutive operators in the agents' private sub-plan, and all their necessary predecessors hold at the current node; and, (**b**) The declaration is "feasible," i.e., it can be achieved by having each of the agents perform at most one action simultaneously with one another $(\bigcup_{n=1}^{l} E_i^l \subseteq A_j^{k+}) \wedge (E_i^* \subseteq F_{\text{ollow}}^n(A_j^{k+})$.

   Thus, each a_i can try to contribute elements of his "next" private subset of constraints to the global set only if they are still relevant and his previous constraints were accepted by the group.

   At the first step, there is only one candidate set for extension — the empty set. Each agent i may declare $E_{g_i}^1$. In our example, the aggregated set of constraints will be $E_{g_1}^1 \cup E_{g_2}^1 \cup E_{g_3}^1 = [\underline{At(c,1)}, Clr(a), \underline{Clr(b)}, Clr(3), At(a_i, gr(a)), At(a_j, gr(b)](=A^1 = A^{1+})$.

   At the second step, a_1 declares E_1^*, which in this example is equal to $E_{g_1}^2 = [\underline{Clr(c)}]$. Similarly, a_2 declares $E_{g_2}^2 = [\underline{On(b,3,V)}]$, while a_3 may already declare $E_{g_3}^3 = [\underline{Clr(d)}]$. All declarations are in $F_{\text{ollow}}^1(\mathcal{A}^1)$, which contains only one subset $(= [\underline{Clr(c)}, \underline{On(b,3,V)}, \underline{Clr(d)}])$.

   At the third step ($A^{2+} = [\underline{Clr(c)}, \underline{On(b,3,V)}, \underline{Clr(d)}]$), a_1 declares $[At(a_i, gr(c))]$ and a_3 declares $[At(a_k, gr(d)), Clr(c)]$.

   At the fourth step, only a_1 has a feasible extension to the current best set, that is $[On(c,1,V)]$ (note that a_3 may not declare his final subgoal before a_2's and a_1's assigned subgoals are satisfied).

   Finally, at the fifth step, only a_3's assigned goal is not satisfied, and he may declare $E_3^* = [\underline{On(d,c,H)}]$.

3. At this step, $\{Ex_r(A_j^{k+})\}$, all the (set-theoretic) maximal feasible extensions of A_j^{k+} with elements of $\bigcup_i E_i^*$, are generated (where each $Ex(A_j^{k+})$ is defined as the following fixed point: $\{I \mid (I \in \bigcup_i E_i^*) \wedge (I \cup Ex(A_j^{k+}) \in F_{\text{ollow}}^n(A_j^{k+})\})$. Note that this is only a subset of the expansions that a "blind planner" should generate.[10]

[10] We can focus only on maximal sets since using the heuristic evaluation they are bound to dominate their

In this simple example, at each step there is only one feasible extension, which is the union of all the agents' declarations.

4. At this stage, based on the extended set of constraints, the agents construct additions to the ongoing candidate plans. Each expansion that was generated in the previous step induces a sequence of operations that achieves it ($P(Ex_r(A_j^{k+}))$). The generation of these sequences is fully described in Section 3.5.

   The corresponding alternative multi-agent plans that have been generated so far are concatenated with the resulting segments of plans: $P(A_j^{k+}) \bowtie \{P(Ex_r(A_j^{k+}))\}$. Thus, the g value of alternative extended sets of constraints is determined. Similarly, the aggregated set A_j^{k+} is replaced by its union with its extensions: $\mathcal{A}^{k+1} = (\mathcal{A}^k \cup \{A_j^{k+} \cup Ex_r(A_j^{k+})\})$.

   At the first step the (sole) extension is fully satisfied by the initial state; therefore, $g(A^1) = 0$. At the second step, $Ex(A^1)$ can best be achieved by $\langle\{Take_1(a,2,0), Take_2(b,4,3)\}\rangle$, yielding a g value of 4 ($= 2 \times 1 + 1 \times 2$). The extension at the second step can be achieved by $\langle Move_1(0,1), Move_2(3,4)\rangle$, with an additional cost of 2. At the third step, the corresponding segment of the multi-agent plan is $\langle Rot_1(c)\rangle$ with additional cost of 4. Finally, the last extension is best satisfied by $\langle Take_2(d,4,1)\rangle$ with a cost of 9. Thus, the overall cost is 19. The entire plan is demonstrated in Figure 4. Notice that the final goal is achieved without any physical contribution on the part of a_3.

5. All expansions are evaluated, in a central manner, so as to direct the search (i.e., find the value, $f' = g + h'$, of the A^* evaluation function):

   (a) The g component of each expansion is simply taken to be the cost of deriving it (the cost of the plan that is induced by the current path plus the additional cost of the multi-agent plan that derives the expansion).

   (b) To determine the heuristic component, h', each agent declares h'_i, the estimate it associates with each newly-formed set of aggregated propositions. This is the cost it associates with completing its "private" sub-plan, given that the expansion is established. The h' value is then taken to be the sum of these estimates ($h'(Ex_r(A^{k+})) = \sum_i h'_i(Ex_r(A^{k+}))$).

   There are two possible policies by which each agent can determine the h' value. According to the first (the "rigid policy"), each agent would "stick" to the initial plan that he generated. The h'_i would then be equal to the cost of completing the plan, given the furthest satisfied point of "his" set of constraints according to that initial plan. A more accurate but also more expensive way (the "flexible policy"), would have the agent reconstruct the plan, given the constraints that have been achieved so far. Note that such a re-planning process will not be too expensive, since from a certain point the new plan would become identical to the initial private plan.

subsets. Alternatively, instead of explicitly calculating these sets it is possible to consider all expansions and let the heuristic evaluation do the pruning.

A^1 is satisfied by the initial state, therefore $f'(A^1)$ is equal to its h' value (that is, the sum of the individual estimate costs, which is 23 $(= 7 + 2 + 14)$). Note that therefore the heuristic evaluation overestimates by 4.

At the second step, the g value becomes 4. According to the rigid evaluation policy, the heuristic estimate, h', would be 17 $(= 6 + 0 + 11)$, while according to the flexible policy $h' = 15$ $(= 5 + 0 + 10)$. a_1's location has become slot O (and not slot 3 as anticipated), while a_3 may reason that a_2 can go to block d's region and therefore update its remaining estimated cost to be only 6 (yielding a total f' of 11).

At the remaining two steps f' remains the same (according to both approaches) since the addition to the g component equals the decrease in the h' component.

6. The aggregated set A_j^{k+} is replaced by its union with all of its extensions: $\mathcal{A}^{k+1} = (\mathcal{A}^k \setminus A_j^{k+}) \cup \{A_j^{k+} \cup Ex_r(A_j^{k+})\}$.

7. The process ends when all "best plans" have been found. Although each plan which is found is sound, the heuristic function is not guaranteed to be an underestimate. Therefore, stopping when the first global plan has been generated may not result in the optimal global plan (i.e., another path, with higher f' value, might actually have a lower overall cost). The system designer needs to establish, as a matter of policy, how much more searching the agents should do (if at all) to discover better global plans. Such policy may be determined experimentally for a particular domain/implementation.

   In the example we stop the search as soon as the goal is achieved—at the fifth step.

3.5 Construction of the Multi-Agent Plan

The multi-agent plan is constructed throughout the process. The construction is made at Step 4 of the algorithm. At this step, all the optimal sequences of operators are determined. We require that the actual plan be constructed dynamically in order to determine the g value of each alternative. Note, however, that it would be possible to instead use Best First Search so as to first determine the entire set of constraints, and only then construct the induced plan. Employing such a technique would still be significantly less time-consuming than global planning.

The construction of the new segments of plans is determined by the cost that agents assign to each of the required actions; each agent bids for each action that each $Ex_r(A_j^{k+})$ implies. The bid is based on the sequence of i's actions in $P(A_j^{k+})$. Thus, the minimal cost sequence, $P(Ex_r(A_j^{k+}))$, is determined.

There is an important tradeoff to be made here in the algorithm, and the decision of the system designer will affect the optimality of the resulting plan. When the agents add on the next steps of the global plan, they could consider the (developing) global plan from its beginning to the current point when deciding on the least expensive sequence of additional steps. This will (eventually) result in a globally optimal plan, but at the cost of having to continually reevaluate the developing plan along the way. Alternatively, it is possible to save all possible combinations of the actions that achieve any $Ex_r(A_j^{k+})$, and thus have a set of plans correspond to each extension. However, since the number of alternatives is exponential with n the space complexity would become much larger. On the other hand, in domains where the sequence of actions is continuous and the cost is identical for all agents (as is the case in our example) the best local decision will also be the best global one, and thus considering only the best induced fraction of plan for each expansion is sufficient.

As a compromise, the agents can instead choose the least expensive sequence of additional steps, considering them only in isolation (i.e., without regard to the parts of the global plan that have already been developed). This resembles a hill-climbing approach that makes the best current decision, without regard to previously explored paths. While the evaluation procedure will be considerably less complex, the resulting global plan can no longer be guaranteed optimal.

An important aspect of the process is that each extension of the set of constraints belongs to the $F^n_{\rm ollow}$ of the already achieved set. Therefore, it is straightforward to detect actions that can be performed in parallel. Thus the plan that is constructed is not just cost efficient, but also time efficient (in effect, in some cases time efficiency may reduce cost efficiency, as can be shown in the example in Section 4.6). In this work, we give primary importance to the cost of the resulting global plan, but if the time of execution is of equal importance, it is possible to maximize the utility of the constructed plan (instead of just minimizing its cost), where utility is a function of execution time *and* cost.

Note that there is another tradeoff to be made here. We let each agent declare $E_i^* \subseteq F^n_{\rm ollow}(\mathcal{A}^{k+})$ to ensure optimality of parallelism in the resulting global plan. However, we can relax this demand, so as to have agents relate just to $F^1_{\rm ollow}(\mathcal{A}^{k+})$ and establish parallelism only after the global plan is fully constructed.

In the example all the suggested methods would result in the same final plan. At the first step the (sole) extension is fully satisfied by the initial state. Therefore, no plan is constructed. At the second step $Ex(A^1) = \{\underline{Clr(c)}, \underline{On(b,3,V)}, \underline{Clr(d)}\}$. These constraints can be achieved by $Take_i(a,2,0)$ and $Take_j(b,4,3)$. The bids that a_1,a_2 and a_3 give to these actions are respectively $[2,5]$, $[4,2]$, and $[6,4]$. Therefore, a_1 is "assigned" to block a and a_2 is assigned to block b while a_3 remains unemployed. The constructed plan is therefore $\langle\{Take_1(a,2,0), Take_2(b,4,3)\}\rangle$, yielding a g value of 4 where both agents perform in parallel.

According to the agents' bids, it turns out in the next step that a_1 should be assigned block c, while a_2 should take care of block d. This work distribution remains the same throughout the rest of the process.

Since subgoals are not assumed to be serializable or independent, sub-plans will tend to interact. There are two kinds of interaction: "positive" interaction relates to operators that serve the same purpose in two different sub-plans, and "negative" interaction denotes conflicting steps. We denote by q a measure of these interactions, such that q grows in direct relation to the number of positive interactions among the sub-plans and in inverse relation to the number of negative interactions. We make no particular assumptions in this paper about q's size. Although we use it to characterize the performance of our algorithm in the theorem below, it is not part of the algorithm itself. It is a domain-dependent parameter, which in practice might be discovered through empirical means, if desired.

Theorem 1 *Given the set $\mathcal{E}_i$ of the sub-plans that achieve the set of instances of subgoals $\{g_i\}$, the merging algorithm will find the optimal multi-agent plan that achieves these subgoals. The process will end within $O(q \times d)$ steps where d is the length of the optimal plan, and q is a measure of the positive interactions between overlapping constraints.*

In comparison to planning by a central planner, the overall complexity of the planning process, $O((n \times b)^d)$, will be reduced to be $O(\max_i b_i^{d_i} + b \times n \times q \times d)$, where in the general case $\max_i b_i^{d_i} \approx \frac{b}{n}^{\frac{d}{n}}$.

Proof:

The underlying idea is that given a function h' that is guaranteed to be at most an overestimate of size δ, the search for an optimal plan should proceed until all candidate plans are greater by δ than the cost of any completed plan. In our case, the overestimate is caused by positive interactions among sub-plans, and q denotes the number of paths needed to be explored to overcome this overestimate.

Let P^* be the optimal multi-agent plan that achieves $\cup_i g_i$, and let $e(G)$ denote the set of constraints that induces P^*. Since given $e(G)$ the optimal set of corresponding operations is found (Step 4 of the process), it is sufficient to prove that $e(G)$ is generated by the process. Since the search continues until the effect of heuristic overestimate dies out, we can assume that the heuristic function is accurate. This assumption makes the A^* search admissible, meaning that the first generated path to the goal is guaranteed to be the optimal one.

The proof is by induction on the subsets of constraints, $E^1(G) \cup E^2(G) \cup \ldots \cup E^d(G)$, that construct $e(G)$ (which in effect equals the number of simultaneous operations that construct P^*).

1. $E^1(G)$; $E^1(G)$ can be achieved from the initial state by simultaneously invoking $r \leq n$ operators. Therefore, each element of $E^1(G)$ belongs to the F^n_{ollow} set of the initial state. On the other hand, $E^1(G)$ must be constructed out of elements of $E^1_i(g_i) \in \mathcal{E}_i(g_i)$ for several $\{a_i\}$. Since, by definition, at the first step each such agent, a_i, will be able to declare the entire set $E^1_i(g_i)$ (including these relevant elements), $E^1(G)$ is guaranteed to be generated.

2. Assume that the claim holds for $E^j(G)_{j=1,\ldots,d-1}$. We have to prove that $E^d(G)$ will be found. According to the induction assumption, $E^{d-1}(G)$ will be reached at some step r. According to the assumption that the h' function is accurate, it will be chosen at step $r+1$ as the best alternative for further expansion (actually, withdrawing the optimality assumption, it will be chosen after at most $r+q$ steps). Since $E^d(G) \models G$ and belongs to $F^n_{\text{ollow}}(E^{d-1}(G))$, it must be the case that each g_i belongs to $F^n_{\text{ollow}}(E^{d-1}(G))$. Thus, each a_i would declare g_i at step r, and $E^d(G)$ will be generated. Thus the entire path, $e(G)$, will be found.

The number of steps in our search is bounded by $O(q \times d)$. At any step of the procedure, the best candidate step is maximally extended. Since we may assume the heuristic function is accurate (for that reason we use the q element as the bound), this set will be chosen at each consecutive step for further expansion. Therefore, the process cannot proceed longer than the number of subsets of constraints needed to achieve G.
The complexity of the search is $O(\max_i b_i^{d_i} + b \times n \times q \times d)$ $(\approx \frac{b}{n}^{\frac{d}{n}} + b \times n \times q \times d)$ where b and d denote the breadth and depth of the search in the central case while b_i and d_i denote the breadth and depth of the search for g_i. Before the merging process begins, each agent generates his individual sub-plan according to his individual perspective. Therefore, as shown by Korf [27], the complexity of this phase is $O(\max_i b_i^{d_i}) \approx \frac{b}{n}^{\frac{d}{n}}$ (see also Section 2.3). As shown above, the merging phase involves $O(q \times d)$ steps. At each step, only feasible extensions are considered. Therefore, in the worst case at each step there will be at most $n \times b$ candidate extensions (that is, we may let the agent evaluate *all* possible extensions [as in the central case] instead of just the relevant ones). Summing up these bounds, we get the above statement. $\square$

3.6 Finding Coherent Instances of Subgoals and Division of Tasks

Thus far, we assumed that a set of coherent instances of subgoals is given and assigned to the different agents. Some problems are *a priori* formulated as grounded instances of subgoals. However, in general, subgoals are specified abstractly (as in the example given in Section 2.1). Although a detailed examination of this problem is beyond the scope of the current research, in this section we briefly describe a heuristic method to derive these instances. The method involves two steps:

1. An iterative search for a set of coherent instances of subgoals ($\{g_i\}$).
2. Coherent binding of soft interface points.

The first step is aimed at finding a set of subgoals that can mutually coexist with respect to $D(G)$ (the specification of the relation between subgoals and the domain axioms). In this step, we care only about unbridgeable conflicts, such as the final resources that are used for the final subgoals. Soft interface points are ignored at this step. The first step is relatively expensive: we have the agents repeatedly generate alternative plans for each subgoal until a set of coherent instances is found (regardless of soft interface points). At the end of each iteration, each agent specifies the instance it has generated for each subgoal. Coherence is then checked. The process continues until a coherent set is found. A natural way to generate the alternative plans is by using a Branch and Bound search.

The second step involves establishing the optimal bindings of the soft interface points based upon the costs of the plans by which they were derived at the first step. We need to determine these bindings so as to find the real g value of alternative (already constructed) plans during the search process. An alternative approach would delay this step to the end of the merging algorithm. Following this approach, we would first construct a global plan in which interface points are not bounded, and only then find the appropriate bindings. This way the binding of soft interface points would become optimal, but the g value of each constructed plan would count on actions that contain uninstantiated variables. Thus, the cost of these actions would be relative to the final binding, and the g value would become an estimate.

Note that it is possible to exchange the order of the two steps. This way, the sub-plans that are generated to achieve the subgoals will be more compatible with the derived global goal. However, the search for instances would become more expensive (since sub-plans will have more possible interactions) and the advantage of planning based on local knowledge would be reduced. Another disadvantage is that the binding itself may not be based on the information that can otherwise be deduced from the sub-plans themselves.

There are several alternative heuristic ways to perform the two steps. In the first step, the simplest policy would assign subgoals to agents arbitrarily. However, if further knowledge is available, such as geographic location (of agents and/or resources) and expertise and capabilities of the agents, there might be justifications for some specific distribution of subgoals (e.g., if agent a_1 has relatively powerful mathematical skills, it should be assigned the subgoal that involves the construction of a mathematical proof). Since such global knowledge is not alway available, we suggest having task division be dependent on the agents' local knowledge. The bindings in the second step may also be determined arbitrarily (within the $D(G)$ constraints), but a preferable way would take the optimal binding with respect to the sub-plans that are generated in the first step.

Another consideration is the capabilities of the different agents. If agents have differing capabilities then, if we want to ensure an optimal global plan, while constructing the plan that achieves a given subgoal an agent should employ operators that are available to other agents. Thus, the complexity of deriving sub-plans will increase.

There are several heuristic ways to speed up the first step, such as assigning disjoint sets of resources prior to the search. Another way is (at each iteration of the search) to have the plans that were generated in the first step constrain (fully or partially) the following step. Such methods would further reduce the optimality of the resulting plan.

After coherent instances are found, the combination that has the minimal estimated cost is chosen, and each agent is assigned one instance (g_i) for the purpose of the merging process.

In the original example in Section 2.1, two possible coherent instances are found after two iterations of the first step, as shown in Figure 5. According to the individual estimates the left one is chosen.

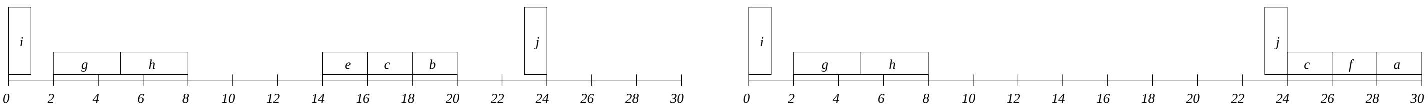

Figure 5: Coherent Combination of Instances

3.7 The Anytime Planning Approach

As described in the previous section, the merging process might be preceded by two heuristic phases in the algorithm (instantiation of subgoals and binding of soft interface points). Because of these heuristics (and the optional heuristic component of the merging process), finding the optimal global plan within our approach is not guaranteed. Obviously, the amount of time and reasoning invested in a search for the optimal plan is not always worthwhile. For that reason, most planners are satisfied with any plan. In this section, we suggest an iterative process for multi-agent planning that heuristically converges to the sub-goal optimal plan. The number of iterations depends on how close to an optimal plan the system designer wishes (or can afford) to get.

Let $t(G)$ denote the time invested in the planning process and $c(G)$ denote the cost (or the utility) of the resulting plan. Then there exists an inverse relation between $t(G)$ and $c(G)$ (i.e., $t(G) \approx \frac{1}{c(G)}$). Let $r(G)$ denote the ratio $\frac{t(G)}{c(G)}$; then (in general) $r(G)$ increases with the time invested in constructing the plan P. Let $r^*(G)$ denote the desired ratio. Our general method can be stated as an iterative process of "generate and merge" sub-plans. At each iteration the resulting global plan is improved.

The underlying idea is an iterative process of modification and validation of sub-plans. Starting with the initial global plan, at each iteration there is an attempt to look for a better global plan. The plan is improved due to "re-examination" of the heuristic decisions made so far (within the merging process itself and its preceding phases). Thus, the general algorithm may be considered as an anytime multi-agent planning procedure [7, 6, 56]. Figure 6 describes the performance profile of the process (the step function lines). The dotted line describes a "diminishing returns" type of decision procedure, to which our process corresponds. As shown, in our case the performance profile is monotonically nondecreasing, instead of monotonically increasing. Up to a certain point in time, there is no suitable plan at all. The $s(P)^*$ line denotes the optimal plan generated for a given subgoal setup, which is in general inferior to the absolute optimal plan described by P^*.

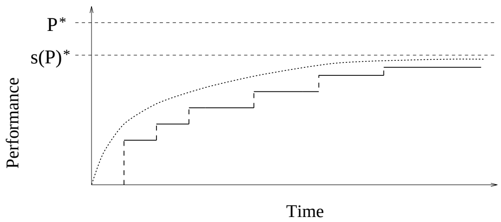

Figure 6: Performance Profile of the Process

Working within the anytime framework, we can accept a sub-optimal initial plan, so as to make sure some plan is generated quickly. This plan will be improved in future iterations. A speed-up in the generation of this plan may be obtained by "freezing" the dimension of the search problem. The freezing is done by imposing rigid constraints on the search. Then these constraints may be relaxed gradually during subsequent iterations. As an example, we may arbitrarily constrain each agent to use a specific separate set of resources, and then relax this demand, or consider only the most general sets of constraints of the sub-plans and then consider the refined sets (i.e., have the sub-plans be generated in a highest level of abstraction that will be reduced in further iterations). Such methods should be heuristically decided by the designers.

4 Executing while Planning for Dynamic Tasks

The multi-agent planning procedure that was presented in the previous section is based on the *incremental* process of merging sub-plans. This attribute of the process makes it very suitable for scenarios where the execution of the actual plan is urgent. In such scenarios it is important that, parallel to the planning process, the agents will actually execute segments of the plan that has been constructed so far [1, 24, 9, 29, 6]. We assume that there is some look-ahead factor, l, that specifies the number of planning steps that should precede the actual execution step(s). We also assume that each agent can construct the first l optimal steps (in terms of constraints) of its own sub-plan. The way individual agents construct their plans is beyond the scope of this paper.

The fact that in order to find the first step(s) of the multi-agent optimal plan it is important for the merging process to have only the corresponding first step(s) of the individual sub-plans, also makes the process very suitable for scenarios where the global goal may change dynamically. In such cases, the required revision of the (merged) multi-agent plan may sufficiently be expressed only by the first l look-ahead steps. Moreover, the process is flexible in response to such global changes, since they may be handled through the division of the new goal into subgoals. Thus, a change in the global goal may be reflected only in changes in several subgoals, and plan revision is needed only in several sub-plans.

In this section, we suggest an approach to the interleaving of planning and execution for dynamic tasks[11] by multiple agents. Agents are dynamically assigned individual tasks that together achieve some dynamically changing global goal. Each agent solves (constructs the plan for) its individual task, then the local plans are merged to determine the next activity step of the entire group in its attempt to accomplish the global goal. Individual tasks may be changed during execution (due to changes in the global goal).

4.1 The Scenario of Task Division for Dynamic Goals

This scenario involves a group $A = \{a_1, \ldots, a_n\}$ of n agents. These agents are to achieve at any given time (t) a global goal (G^t, which may change over time). Following the distribution of individual planning tasks ($\{g_1^t, \ldots, g_m^t\}$), each agent determines the first l look-ahead steps of p_i^t, the plan (expressed as a set of constraints) that accomplishes its present task(s). Then,

[11]A task is an instantiated subgoal, with an associated time stamp. While a subgoal assignment is done once for the entire process, an agent may periodically be assigned new tasks, and abandon old tasks.

based on the individual plans, the agents communicate as they construct the next l multi-agent steps towards the achievement of the global goal. During this interleaved process of planning and execution, some agents may be assigned different tasks. As before, we will assume, without loss of generality, that $m = n$.

4.2 Assumptions and Definitions

We here adjust the notation that was used in Section 3.1 to the case of interleaved planning and execution:

- The *global goal* at time t, G^t, is a set of predicates, possibly including un-instantiated variables. G^t is divided into n tasks $[g_1^t, g_2^t, \ldots, g_n^t]$. Each task is a set of instantiated (grounded) predicates such that their union satisfies G^t ($\cup_i g_i^t \models G^t$). Tasks are assigned to individual agents only for the purpose of the planning process; the actual execution is divided differently.

- Any plan that achieves g_i^t is denoted by p_i. Each such sub-plan is a sequence of operators $\langle op_1^t, \ldots, op_k^t \rangle$ that, invoked in the current configuration of the world s^t, establishes g_i^t.

- $\hat{e^t}_i(g_i^t)$ is the set of absolutely necessary general constraints needed for any plan to achieve the grounded instance of the subgoal g_i^t, starting with some initial state S^t. In accordance with the partial order over these constraints, we divide $\hat{e^t}_i(g_i^t)$ into subsets of constraints. Each such subset within $\hat{e^t}_i(g_i^t)$ 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 these subsets is denoted by $\mid \hat{e^t}_i(g_i^t) \mid$. $\mathcal{E}_i^t(g_i^t)$ denotes the set of all the refinements of $\hat{e^t}_i(g_i^t)$.

- The look-ahead factor l denotes the number of planning steps that should precede each actual execution phase of the multi-agent plan in progress. Similarly, each individual look-ahead factor, (the horizon of each individual sub-planning process), is denoted by l_i. We assume that $\min_i l_i \geq l$.

4.3 An Example

As an example, consider another simple scenario in the slotted blocks world as described in Figure 7. We use the same operators and predicates as described in Section 2.1. In this scenario, there are two agents (a_1, a_2) and 3 blocks (a,b,c) with a length of 2 feet each.

The initial state is:
$\{At(a_1,1), At(a_2,9), At(a,12), On(a,12,H), At(c,12), On(c,a,H), At(b,4), On(b,4,H)\}$.

The agent's initial tasks are (respectively):
$g_1^0 = \{At(a,16), At(b,16), On(a,b,H)\}$ and
$g_2^0 = \{At(b,16), At(c,16), On(b,c,H)\}$.

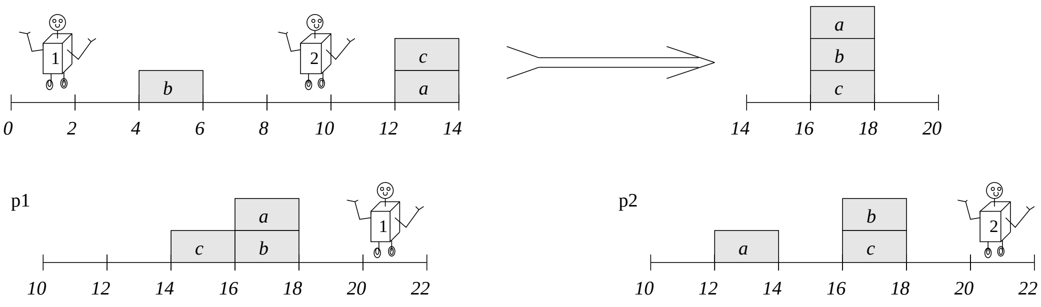

Figure 7: A Simple Slotted Blocks World Example

The combination of these two tasks will eventually establish the global goal $G^0 = \{At(a,16), At(b,16), On(a,b,H)\}$. However, as shown in the lower part of Figure 7, the two corresponding individual sub-plans would result in two inconsistent final configurations (which do not satisfy the final goal).[12]

After 3 execution time-steps the global goal is changed to be $G^3 = \{At(a,16), On(a,c,H)\}$. Accordingly, task g_1^0 is changed to be $\{At(a,16), On(a,c,H)\}(= g_1^3)$ and g_2^0 is canceled.

Given these (changing) tasks, our agents are to go through a planning process that will result in satisfying the final goal (discussed below).

4.4 The Main Process for Interleaved Planning and Execution

As in the previous scenario, the key element of the approach is a cost-driven merging process that results in a coherent global plan (of which the first l sets of simultaneous operators are most relevant), given the sub-plans. We assume that at each time step t each agent, i, has been assigned (again, only for the purposes of the *planning process*) one task. Given that instance, the agent will derive (the first l steps of) p_i^t, the sub-plan that achieves it. Note that once i has been assigned g_i^t at any given t, the plan it derives to accomplish the subgoal stays valid (for the use of the algorithm) as long as g_i^t remains the same. That is, for any time $t+k$ such that $g_i^{t+k} = g_i^t$, it holds that $p_i^{t+k} = p_i^t$. Therefore, re-planning can be modularized among agents; one agent may have to re-plan, but the others can remain with their previous plans.

As before, the process is iterative. The underlying idea is the *dynamic generation of alternative execution steps* that identifies the optimal global plan and thus the next l optimal global execution steps. At each step, all agents state additional information about the sub-plan of which they are in charge. The next l optimal steps are then determined and the current configuration of the world, s^t, is changed to be s^{t+1}. The process continues until all tasks have been accomplished (the global goal as of that specific time has been achieved). Plans are represented as sets of absolutely necessary sets of constraints that enable them.

[12]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 [48].

Note that when we refer to an individual agent's sub-plan, we are really only concerned with that initial part of the sub-plan that is relevant to the merging process at any specific time step t (which, in turn, depends on the global look-ahead factor). The way each individual determines this initial part of his sub-plan (i.e., what to do next) under time constraints is beyond the scope of this paper.

Essentially, the search method employs a Hill-Climbing algorithm. The heuristic function (h') that guides the search is dynamically determined by the agents during the process. As in the previous procedure, the global heuristic value of each alternative expansion, h', is taken to be the sum of the approximate remaining costs that each agent assigns to his own sub-plan ($\sum_i^n h'_i$). This heuristic evaluation suffers from the same drawbacks that were described in Section 3.2, and therefore the resulting multi-agent plan may only approximate the actual optimal global plan. However, this heuristic function is powerful enough to ensure only limited effect of the classic problems of Hill Climbing (foothills, plateaus, and ridges). The reason is that at each time step the misjudgment of the heuristic function is bounded by the effect of "positive" and "negative" interactions among the sub-plans, which (for the purpose of evaluation) offset one another. Given that the deliberation time that may precede each execution step is limited, the accuracy of the decision (based on the heuristic function) is relatively very satisfactory. Moreover, the longer the time allowed for deliberation (the look-ahead factor) is, the more accurate the decision will become.

4.5 The Algorithm for Parallel Planning and Execution

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 constraints that has induced the plan executed thus far.

The set of constraints that has induced the multi-agent plan that has been executed up to time step t, P^t, is denoted by $\mathcal{A}^t$. $\mathcal{A}^{t+}$ denotes the set of all aggregated sets of constraints at step t.

We now continue the example that we began in Section 4.3. To simplify things we will use throughout this example only $\hat{e}_i$ instead of $\mathcal{E}_i$. We also use a look-ahead factor of one ($l = 1$) and assume that l_i is not bounded (i.e., the sub-plans are fully generated so as to accomplish the final subgoals, but each generation of global expansion will follow an immediate execution step of the best induced fraction of the global plan). The resulting steps of the procedure are illustrated in Figure 8.

The exact procedure is defined as follows:

1. At step 0 each agent i finds $\mathcal{E}_i^0(g_i)$—the set of all alternative absolutely necessary temporally ordered sets of constraints that achieve the task g_i^0, starting from the initial state S^0, and its refinements. Actually, the agent has to determine only the first l_i subsets of this set depending on the look-ahead factor.

   The set of alternatives is initialized to be the empty set ($\mathcal{A}^0 = \emptyset$).

In our example, we have (using the same notation as in Section 3.4): $\hat{e^0}_1(g_1) = \{[Clr(b), Clr(c)]^{[0]} \cup [At(a_i, gr(b)), Clr(b), Clr(c)]^{[3]} \cup [Clr(b), Clr(c), \underline{At(b,16)}]^{[24]} \cup [At(a_i, gr(c)), Clr(b), Clr(c), \underline{At(b,16)}]^{[4]} \cup [Clr(a), Clr(b), \underline{At(b,16)}]^{[4]} \cup [At(a_i, gr(a), Clr(a), Clr(b), \underline{At(b,16)}]^{[2]} \cup [\underline{On(a,b,H)}, \underline{At(a,16)}, \underline{At(b,16)}]^{[8]}\}$ (this ordered set induces the plan $\langle Move_1(1,4), Take_1(b,4,16), Move_1(16,12), Take_1(c,12,x_c = 14 \mid 10), Move_1(x_c,12), Take_1(a,12,16)\rangle$ with a total cost of 45). and $\hat{e^0}_2(g_2) = \{[Clr(c), Clr(b)]^{[0]} \cup [At(a_j, gr(c), Clr(c), Clr(b)]^{[3]} \cup [Clr(b), \underline{At(c,16)}, Clr(c)]^{[8]} \cup [At(a_j, gr(b), Clr(b), \underline{At(c,16)}, Clr(c)]^{[12]} \cup [\underline{At(b,16)}, \underline{At(c,16)}, \underline{On(b,c,H)}]^{[24]}\}$ (inducing the plan $\langle Move_2(9,12), Take_2(c,12,16), Move_2(16,4), Take_2(b,4,16)\rangle$, with a cost of 47). Note, that $\mathcal{E}_i$ has in general, even in this example, more elements. For example, $\mathcal{E}_1$ would also include a different order, in which block c is removed from the top of block a before block b is placed at region 16, and $\mathcal{E}_2$ would include the refinement of e_2 in which, prior to removing block c, block b is moved to region 14 (inducing a_2's optimal plan: $\langle Move_2(9,4)$,$Take_2(b,4,14)$,$Move_2(14,12)$, $Take_2(c,12,16)$,$Move_2(16,14)$,$Take_2(c,14,16)\rangle$).

2. At step t, each agent declares $E_i^{A_j^t} \subseteq E_i^{l+1}$ only if, for all $k \leq l$, E_i^t is already modeled by P^t (i.e., all its predecessors in the private sub-plan were already declared and merged into the global plan), and the declaration is "feasible," i.e., it can be reached by invoking at most n operators simultaneously on that set: $(\bigcup_{n=1}^{l} E_i^l \subseteq A_j^t) \wedge (E_i^{A_j^{t+}} \subseteq F_{\text{ollow}}^n(A_j^t))$.

   At the first step, there is only one candidate set for extension — the empty set. Each agent i may declare $E_{g_i}^1$. In our example, this will be $E_{g_1}^1 = E_{g_2}^1 = [Clr(c), Clr(b)]$.

   At the second step, $A^1 = [Clr(c),\ Clr(b)]$, a_1 declares $E_1^{A_1^{1+}}$, which in this example is equal to $E_{g_1}^2 = [At(a_i, gr(b)),\ Clr(b),\ Clr(c)]$. Similarly, a_2 declares $E_{g_2}^2 = [At(a_j, gr(c), Clr(c),\ Clr(b)]$. Both are in $F_{\text{ollow}}^2(\mathcal{A}^1)$, which contains only one subset.

   At the third step $A^2 = [At(a_i, gr(b),\ At(a_j, gr(c),\ Clr(c),\ Clr(b)]$, a_1 declares $[\underline{At(b,16)}, Clr(a), Clr(b)]$ (counting on the simultaneous execution of $\langle\{Take_j(c,12,10 \mid 14), Take_i(b,4,16)\}\rangle$), and a_2 declares $[\underline{At(c,16)},\ \underline{On(b,c)}]$ (counting on the simultaneous execution of $\langle\{Take_j(c,12,16 \mid 14),\ Take_i(b,4,16)\}\rangle$).

   If the global goal would have remained the same, in the fourth step a_1 would be able to fully declare his final task ($[\underline{On(a,b,H)}]$) and the global goal G^0 would be achieved. However, at this step the global goal changes to be $\{At(a_i, gr(b)), Clr(b), At(a_j, gr(a)), Clr(a)]\}$. e_1^0 is no longer valid, and is replaced by $e_1^3 = \{[At(a_i, gr(b)),\ Clr(b),\ At(a_j, gr(a)),\ Clr(a)]^{[0]} \cup [Clr(a), Clr(c)]^{[4]} \cup [At(a_i, gr(a)), Clr(a), Clr(c)]^{[2]} \cup [\underline{At(a,16)}, \underline{On(a,c,H)}]^{[8]}\}$ (inducing the plan $\langle Take_i(b,16,14),\ Move_i(14,12),\ Take_i(a,12,16)\rangle$). Therefore, a_1 declares the set $E_1^{A^3} = [At(a_i, gr(b)),\ Clr(b), At(a_j, gr(a)),\ Clr(a)]$ (which is in $F_{\text{ollow}}^2(A^3)$) while a_2 has no task at all.

   At the fifth step a_1 declares $[\underline{On(a,c,H)}, \underline{At(a,16)}]$, which satisfies the final goal.

3. At this step, $\{Ex_r(A_j^t)\}$, all the maximal feasible extensions of A_j^t with elements of

$\bigcup_i E_i^{A_j^{t+}}$ are generated (where each $Ex(A_j^t)$ is defined as the following fixed point: $\{I \mid (I \in \bigcup_i E_i^{A_j^{t+}}) \wedge (I \cup Ex(A_j^t) \in F_{\text{ollow}}^n(A_j^t)\})$.

At the first step, the aggregated set of constraints is $[Clr(c), Clr(b)]$.

At the second step, both declarations may coexist consistently, and there is therefore only one successor to the previous set of constraints: $A^2 = A^1 \cup$ $[At(a_i, gr(b), At(a_j, gr(c), Clr(c), Clr(b)]$.

At the third step, A^2 has only one extension: $[\underline{At(b,16)}, Clr(a), Clr(b), \underline{At(c,16)}, \underline{On(b,c)}]$. Notice that if we would allow agents to only declare $F_{\text{ollow}}^1(A^2)$, then at the third step, A^2 would have four possible extensions: $Ex_1 = [At(c,16), Clr(b), Clr(c)]$, $Ex_2 = [At(b,16), Clr(b)\ , Clr(c)]$, $Ex_3 = [At(b,16), Clr(b), At(c,16)]$ (where b is located on a), and $Ex_4 = [At(b,16), Clr(b), At(c,16)]$ (where a is located on b). Thus, $\mathcal{A}^3$ would have four different members (the union of A^2 with each of the extensions).

At the fourth step, A^3 has one extension, $[At(a_i, gr(b)), Clr(b), At(a_j, gr(a)), Clr(a)]$, which includes both agents' declarations.

At the fifth step, the sole extension of A^4 is the description of both tasks at time 4.

4. All extensions are evaluated so as to choose the next step of the multi-agent plan (i.e., find the h' value of the Hill-Climbing search). Each agent declares h'_i, the estimate it associates with each newly-formed set of aggregated constraints (the cost of completing "his private" task given that set). The h' value is then taken to be the sum of these estimates $(h(Ex_r(A^{t+})) = \sum_i h'_i(Ex_r(A^{t+})))$.

   A^1 is satisfied by the initial state, therefore, $h(A^1)$ is equal to its h' value (that is, the sum of the individual estimate costs, which is 92 $(= 45 + 47)$). This value changes at the second step to 82 since both agents carry out the first step of their initial plans.

   At the third step, the heuristic value becomes $12 = ((4+8)+0)$ according to both the rigid and flexible evaluation policies (at this step, the heuristic evaluation is accurate). After two additional steps, the initial global goal would be achieved with a total cost of 46, meaning that the initial heuristic evaluation was an overestimate of 46.

   However, since the goal changes, the new heuristic value becomes 14. At the fourth step the heuristic estimate becomes 8, which is the accurate cost. The final goal is finally achieved with a total cost of 16, meaning that the initial heuristic evaluation of the second global goal was an underestimate of 4.

5. Each expansion induces a sequence of operators that achieves it $(P(Ex_r(A_j^{t+})))$. At this stage, based on the best alternative extension, $Ex_r^*(A_j^{t+})$, that was chosen in the previous step, the agents *execute* additions to the ongoing executed plan. The generation of these sequences is fully described in Section 4.6.

For $l > 1$, the actual execution is delayed by l iterations. In this case the heuristic evaluation will take into consideration also the estimated cost of deriving each extension (as described in the algorithm in Section 3.4).

6. The process ends when the "current" global goal has been achieved.

   In the example, the process stops as soon as the goal is achieved—at the fifth step.

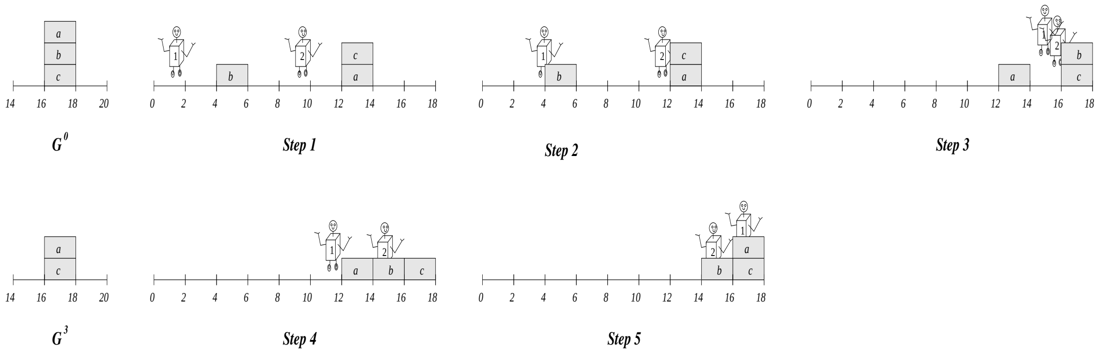

Figure 8: Steps of the Interleaved Planning and Execution Process

4.6 Construction and Execution of the Multi-Agent Plan

The actual multi-agent plan is constructed and executed throughout the process. The construction is made at Step 5 of the algorithm. At this step, all the optimal sequences of operators are determined. The construction of the new segments of plans is determined by the cost that agents assign to each of the required actions; each agent bids for each action that $Ex^*_r(A^{t+}_j)$ implies. The bid is based on the sequence of i's actions in $P(A^t_j)$. Thus, the minimal cost sequence, $P(Ex^*_r(A^{t+}_j))$, is determined.

There is an important tradeoff to be made here in the algorithm, similar to the one described above in Section 3.5. Since the agents choose the least expensive sequence of additional steps, considering them only in isolation, and make the best current decision, the resulting global plan cannot be guaranteed optimal. The effect of this drawback might be reduced by increasing the time spent on planning at the expense of delaying execution (this is, in fact, the look-ahead factor mentioned above). Having the agents plan larger segments of the plan before actually executing it will improve the eventual result, but on the other hand will complicate the search, and delay execution (which may be undesirable).

In the example we get the same final plan regardless of the look-ahead factor.

At the first step the (sole) extension is fully satisfied by the initial state. Therefore, no plan is constructed (or executed).

At the second step $Ex(A^1) = [At(a_i, gr(b), At(a_j, gr(c), Clr(c), Clr(b)]$. These constraints can be achieved by $Move_i(gr(a_i), gr(b))$ and $Move_j(gr(a_j), gr(c))$. The bids that a_1 and a_2

give to these actions are respectively $[3, 12]$ and $[6, 3]$. Therefore, a_1 is "assigned" to block b and a_2 is assigned to block c. The executed plan is therefore $\langle\{Move_1(1,4), Move_2(9,12)\}\rangle$.

Based on the agents' bids, the chosen extension in the third step can be achieved by: $\langle\{Take_1(b,4,16), Take_2(c,12,16)\}\rangle$ and this fraction of the plan is executed.

At the fourth step, the extension is achieved by $\langle\{Move_i(16,12), Take_j(b,16,14)\}\rangle$. Either agent may be assigned each task. Notice that this plan is more expensive than a serial one-agent plan. This phenomenon shows the disadvantage of the parallelism with respect to cost efficiency.

The final extension is established by $\langle Take_i(a,12,16)\rangle$. The corresponding final plan that has been executed is: $\langle\{Move_1(1,4), Move_2(9,12)\}, \{Take_2(c,12,16), Take_1(b,4,16)\}, \{Move_1(16,12), Take_2(b,16,18)\}, Take_1(a,12,16)\rangle$.

Theorem 2 *Let the cost effect of "positive" interactions among members of some subset, p, of the set of sub-plans, P^t, that achieves G^t be denoted by δ_p^+, and let the cost effect of "negative" interactions among these sub-plans be denoted by δ_p^-. Accordingly, let $\delta = \max_{p\in P^t} \mid \delta_p^+ - \delta_p^- \mid$.*[13] *We say that the multi-agent plan that achieves G^t is δ-optimal, if it diverges from the optimal plan by at most δ.*

Then, at any time step t, employing the merging algorithm, the agents will follow a δ-optimal multi-agent plan that achieves G^t.

Proof:

Consider the current world state s^t, at time step t (that state was derived by the multi-agent plan that was executed prior to time t). Let A^s be the set of constraints that induces s^t (actually this set is the description of s^t), and let G^t be the global goal at that time step. We have to show that the next l steps generated at that step do not diverge by more than δ from the actual optimal plan.

Since larger look-ahead factors have a correcting effect on the heuristic evaluation, we need only consider $l = 1$.

By allowing a deviation of the plan from the optimal one within the bound of the combined effect, δ, we can consider the heuristic function to be accurate.

Let $E^1(G^t)$, denote the first subset of the constraints that induces any optimal multi-agent plan that derives G^t, given s^t. Since the heuristic evaluation is (assumed to be) accurate, it is guaranteed the if $E^1(G^t)$ will be generated it will be chosen for execution (Step 4 of the algorithm). Furthermore, given any extension, $Ex_r(A^s)$, of the set of constraints that is satisfied by s^t (namely A^s), the corresponding optimal sequence of operations that achieves it is found (Step 5 of the process). Thus, if $E^1(G^t)$ will be generated, the corresponding optimal sequence of operators will be found. Therefore, it is sufficient to show that $E^1(G^t)$ will indeed be generated.

$E^1(G^t)$ should be achieved from A^s by simultaneously invoking $r \leq n$ operators. Therefore, each element of $E^1(G^t)$ belongs to $F^n_{\text{ollow}}(A^s)$. Thus, any agent that needs to declare any element of $E^1(G^t)$ can do so.

[13] The effect of heuristic overestimate (due to positive future interaction between individual plans) and the effect of heuristic underestimate (due to interference between individual plans) offset one another, as discussed above.

On the other hand, $E^1(G^t)$ would score the highest heuristic value (h' is accurate). That is, it should partially satisfy some of the agents' declarations, $E_i^{A_j^t}$, at step t. But for that same reason, these agents will find it necessary to declare these declarations. $\square$

5 Self-Motivated Agents with Individual Goals

The fact that the process of merging sub-plans is guided by individual estimates, in a manner that resembles a vote, makes our technique suitable to scenarios where each agent has its own *individual* goal.

With our approach, coordination of actions by a group of agents corresponds to a group planning process. 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.

The primary consideration of the process that is presented below is not the complexity of the procedure, but rather the quality of the final outcome (however see Section 5.8).

5.1 The Scenario of Agents with Private Goals

This scenario involves a group of n agents operating in a world currently in state s_0. Each agent has its own private goal (g_i). 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 ($w_i(g_i)$). 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 in Section 5.4).

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.

5.2 An Example

Consider a simple scenario in the classic 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: **On**(Obj_1, Obj_2)—Obj_1 is stacked onto Obj_2; **Clear**(Obj)—there is no object on Obj; and **At**$(Obj, Slot)$—Obj is located at $Slot$. The function **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 $On(1, b)$).

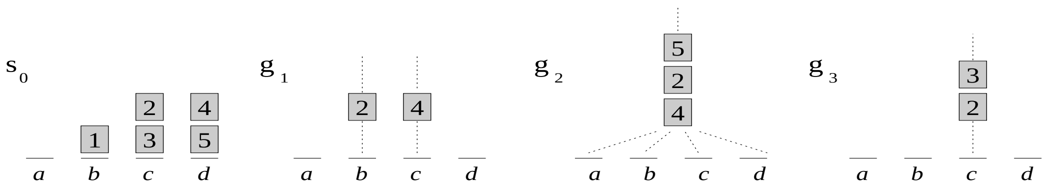

Figure 9: A Blocks World Example of Individual Goals

There are three agents operating in the world. The start state is shown at the far left of Figure 9. 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 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:
[$Prec: Clr(Obj_1), Clr(Obj_2), On(Obj_1, Obj_x)$],
[Del: $On(Obj_1, Obj_x), Clr(Obj_2), At(Obj_1, loc(Obj_1))$],
[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.

5.3 The Concept of Solution

In this paper we are interested in consensus states that maximize the welfare of the entire society.[14] Assuming that each agent has a utility function over possible sets of constraints (see Section 5.4), we define the following criteria for global maximization of utility:

- $c(s_0 \rightsquigarrow E)$ is the minimal cost needed to satisfy the set of constraints E given s_0, using any combination of (multi-)agent actions. $c_i(s_0 \rightsquigarrow E)$ is i's share of the work.

- Let $\mathcal{E}$ denote all possible sets of the domain's constraints. Each agent a_i has a utility function $u_i : \mathcal{E} \rightarrow \mathbb{R}$, over sets of satisfied constraints. Essentially, this utility is some function of $w_i(g_i)$, the worth that i assigns to its final private goal, and possibly other elements (see Section 5.4).

  Given any set of individual utility functions $\{u_i\}$, of the agents in the decision group, we define the Social Welfare/Utility $U(E)$, of any set of constraints E to be a function of all the individual worth functions, minus the cost of achieving the set from s_0; $U(E) = -c(s_0 \rightsquigarrow E) + f^U(E)$, where $f^U : \mathcal{E} \times u_1 \times u_2 \times \ldots \times u_n \rightarrow \mathbb{R}$ is the global utility function based on the individual utility functions.

[14]Issues regarding solution criteria are discussed in [8], in the context of combining various default theories.

There are many ways to measure global utility, and it is not obvious how environment designers will decide on one or another. Considerations other than pure utility values (such as income and fairness) might need to be taken into account. For example, it might be desirable in some scenarios to look for a state that maximizes the median of the utilities or some weighted sum of these utilities, or, following an egalitarian approach [42] we might want to maximize the minimal utility ($\max \min_i u_i$) or minimize the differences in utility gain ($\min \max_{i,j} \mid u_i - u_j \mid$). Our procedure here is not dependent on any particular function, and is equally suitable regardless of the one chosen.

- We say that a decision procedure satisfies the Social Welfare Criterion if it ends up choosing a nonempty set of consistent sets of constraints $SW(\mathcal{E})$ such that $\forall E[E \in SW(\mathcal{E}) \Rightarrow U(E) \geq max_{E' \in \mathcal{E}} U(E')]$ (i.e., SW(E) contains the states with maximal social utility).
- Given f^U and $\{u_i\}$ we define the gap bound ∇_{f^U} to be the maximal gap between any local maximum of the function f^U and the local minimum that follows it.[15]

Our procedure below searches for a set of constraints that satisfies the Social Welfare Criterion. Our technique exploits agents' willingness to be partially satisfied by many states, while attempting to reach their full goal states. This concept of partial satisfaction is reminiscent of the "soft goals" approaches taken in [31]. Note, however, that even though we are searching for a compromise state, the process is still appropriate even if only fully satisfied goals have any value (as, for example, in [21]). In that case, all worth functions are step functions, and ∇_{f^U} would be relatively large.

5.4 The Assessment of a Single Set's Worth

Evaluation of sets of constraints plays an important role in the search procedure. In the two previous multi-agent planning procedures, we have assumed that all agents will assess the worth of a given set of constraints (each possible extension in the merging process) according to the estimated additional cost that they associate with the (yet) unsatisfied part of their individual constraints. In the current scenario, however, the agents are not assumed to have one common goal. Instead, each agent has it own private perspectives and considerations in assessing a set's worth. In this section we consider some plausible approaches for determining this worth.

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.

[15] This definition determines the maximal possible drop in the global utility along the search process, and serves the pruning process embedded in our algorithm.

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.[16]

There are several ways to refine the worth function so as to solve this problem. One way is by making the "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 [23] and [25] 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. For such functions $\nabla_{f^U} \leq 0$. As an example, it would be sufficient to assume above that $\forall i (c(A_1 \rightsquigarrow A_2) \leq c_i(A_1 \rightsquigarrow A_2))$.

5.5 General Overview of the Merging Process for Private Goals

The algorithm enables a group of agents to find the state that maximizes their social welfare function f^U. Similarly to the previous procedures, 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.

However, since agents have their own individual utility functions ($\{u_i\}$) the resulting evaluation (the global heuristic function) is not powerful enough to guide the search according

[16] This is reminiscent of the Little Nell planning paradox, where an agent solves his problem by deriving a suitable plan, but has no need to carry it out once he identifies that it indeed solves his problem [35].

to classic Hill-Climbing or A^* (as used before). Instead we use a variant of these methods. In effect, the search method in this algorithm is a kind of *parallel* A^* search. For example, the value of a state (its social utility) is taken to be the sum of individual estimates of the distance to the goal state (the h' component) and the actual cost of achieving it (the g component). But parallel searches are carried out simultaneously in each promising direction (wherever social utility does not decrease — reminiscent of Hill-Climbing). When the search encounters a direction where social utility decreases beyond the gap-bound limit, ∇_{f^U}, the search is terminated (that path is pruned).

The procedure has the following advantages, in addition to those advantages described in previous sections: **(a)** 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 [13]); **(b)** agents are required to submit only the minimally "conflict-sufficient" information about their goals (described below), further maintaining their privacy.

The description of the process makes use of the following additional definitions:

- 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 ($I \wedge \bar{I} \models F_{\text{alse}}$).[17]

- We say that a set of apparently conflicting constraints E is semi-consistent ($E \not\models_{\text{tmp}} F_{\text{alse}}$) if the removal of temporary constraints makes it consistent (see Section 6 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.

5.6 The Algorithm for Private Goals

This section describes the algorithm in more detail. 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 $\mathcal{A}^{k+1}$ before it is pruned (similarly, its non-pruned components are denoted by A_j^{k+}).

The exact procedure is defined as follows.

[17] In the two previous scenarios there was no need to deal explicitly with temporary constraints, since the subgoals were decomposed out of one coherent final goal. In this scenario, on the other hand, the final subgoals are likely to be in conflict. Our treatment of temporary constraints allows us to delay, as much as possible, such conflicts.

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 set of alternatives is initialized to be the empty set ($\mathcal{A}^0 = \emptyset$, its induced set of states is s_0).

2. At step k, each agent may declare $E_i^{A_j^k} \subseteq E_i^{l+1}$ only if E_i^l 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: $\exists A_j^k[(A_j^k \in \mathcal{A}^k) \wedge (\bigcup_{n=1}^{l} E_i^l \subseteq A_j^k) \wedge (E_i^{A_j^k} \subseteq F_{\rm ollow}(A_j^k))]$.

   i is allowed to try to enforce 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. Notice that it is in i's best interest to give a true declaration, since bringing about E_i^{j+1} is useless without establishing E_i^j first. Such insincere declarations can also be easily tracked by the other agents, since an insincerely declared constraint will not be used subsequently.

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 $\{A_j^k \cup \{I \mid (I \in \bigcup_i E_i^{A_j^{(k+1)+}}) \wedge ((I \cup A_j^{(k+1)+}) \not\models_{\rm tmp} F_{\rm alse})\}\}$).

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.

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, $\mathcal{A}^{k+1} = \{A_j^{k+1} \mid A_j^{k+1} \in \mathcal{A}_j^{(k+1)+} \wedge \forall l \leq k[U(A_j^{k+1}) \geq U(A_j^l) - \nabla_{f^U}]\}$.

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.

7. Given the chosen set of constraints, the actual multi-agent agent plan that achieves it is constructed incrementally by having the agents bid over its induced operator. This bidding phase is done in a manner similar to the one described in Section 3.4.

Theorem 3 *Given any set $\{u_i\}$ of individual worth functions, and any global utility function f^U and the corresponding gap bound ∇_{f^U}, this mechanism finds all consistent sets that satisfy the Social Welfare Criterion.*

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 (the order of the length of the plan that *derives the most distant state that answers the Social Welfare Criterion, from the initial state).*

Proof:

The general idea is that each set of aggregated constraints is extended by the search process until the "limit of consistency"; therefore, any maximal consistent set of aggregated constraints will eventually be reached. On the other hand, it is the case that any state that answers the Social Welfare Criterion is induced by some maximal set of consistent constraints. Therefore, the search process will reach sets of constraints that induce all Social Welfare states.

For simplicity, we assume that $\nabla_{f^U} = 0$. The generalization of the proof to worth functions for which $\nabla_{f^U} > 0$ is straightforward.

Let $A \in SW(\mathcal{A})$ and let K be the last step. We need to prove that $A \in \hat{\mathcal{A}}^K$ (the final set of states in step 6 of the algorithm above).

We will prove a stronger claim: *for any given (imposed) n, the procedure finds all sets, $\hat{A}$ such that $l(P(s_0 \rightsquigarrow \hat{A})) \leq n$ and $\hat{A}$ answers the social welfare criterion with regard to that additional constraint.*

Let $\mathcal{A}\rceil^n$ denote all sets that are n-distant (or less) from s_0. Since, given n, each such $\hat{A}$ is in $SW(\mathcal{A}\rceil^n)$, we have (by the definition of SW for any other set $A' \in \mathcal{A}\rceil^n$, $A' \neq \hat{A}$) that $f^U(\hat{A}) \geq f^U(A')$. Therefore, if $\hat{A}$ is a member of any $\mathcal{A}^k\rceil^n$ it will be a member of $\hat{\mathcal{A}}^K\rceil^n$. It is thus sufficient to prove that for some k, $\hat{A}$ belongs to $\mathcal{A}^k\rceil^n$. Because of step 4 in the procedure, it is sufficient to prove (for any given n) that $\hat{A}$ is generated at some stage k of the procedure. Or, using the procedure's notation: $\exists k[\hat{A} \in \mathcal{A}^{k+}$. The proof is by induction on n:

1. $n = 1$:

   (a) $l = 0$, therefore, $\hat{A} = \emptyset$. From the procedure it follows that $\hat{A} \in \mathcal{A}^0$ (which is set to be the empty set).

   (b) $l = 1$; $\hat{A}$ can be achieved from s_0 by one operator. $f^U(s_0) \leq f^U(\hat{A})$, so it must be the case that $\hat{A}$ is preferred to s_0 by some agents $\{a_i\}$ (i.e., for each such agent it holds that $u_i(\hat{A}) \geq u_i(s_0)$). To win (score) a higher worth, an aggregated set of constraints must satisfy more constraints of the individual sets (we assume $\nabla_{f^U} = 0$).

   Denote these additional constraints by E^+. It follows that $E^+ \subseteq \cup_i E_i^{\emptyset}$ (a subset of the declarations of the agents that prefer $\hat{A}$ to s_0).

   Since s_0 enables E^+ ($E^+ \in f_{\text{follow}}(s_0)$), then actually $E^+ \in \cup_i E_i^2 \subseteq \mathcal{A}^{1+}$ (this is because agents want to maximize their own utility, and thus impose as many constraints as they can at each step). Therefore, $E^+ \in \mathcal{A}^1$ and E^+ is the required $\hat{A}$.

2. Assume that the claim holds for any $l \leq n-1$ and that $l(P(\hat{A})) = n$. Let $\check{\mathcal{A}}$ denote all the sets that could precede $\hat{A}$, i.e., sets of constraints $\check{A}$ such that $[\check{A} \subset \hat{A}] \wedge [\check{A} \cup f_{\text{ollow}}(\check{A}) = \hat{A}] \wedge [l(P(\check{A})) \leq n-1])$.

   For each set, $\check{A}$, in $\check{\mathcal{A}}$, it holds that $f^U(\check{A}) \leq f^U(\hat{A})$. It must then be the case that $\hat{A}$ satisfies more constraints than any such $\check{A}$. Let $E^+(\check{A})$ denote the set of these additional constraints.

   From the monotonicity of the worth functions with regard to satisfied constraints ($\nabla_{f^U} = 0$), it follows that there must be at least one set, $\check{A}'$, in $\check{\mathcal{A}}$ (consistent or semi-consistent) that is a member of $SW(\mathcal{A}\rceil^r)$ (where $r = l(P(s_0 \rightsquigarrow \check{A}')))$. $\hat{A}$ is superior to that set since it satisfies $E^+(\check{A}')$, but the only way for these additional constraints to be satisfied is by invoking some operators in $\check{A}'$. That is, given the restriction that plans must not exceed length r, $\check{A}'$ is a maximizer of the social welfare.

   On the other hand, since $\check{A}' \in SW(\mathcal{A}\rceil^r)$, it follows from the induction assumption that any such $\check{A}'$ will be found by the procedure at some step $\check{k}$. Also, since $\check{A}' \in SW(\mathcal{A}\rceil^r)$, it does not reduce the social utility of any of its preceding sets, and thus $\check{A}'$ is not pruned in the $\check{k}$th step.

   Let G denote the group of agents that prefer $\hat{A}$ to $\check{A}'$. For each member of G, $u_i(\hat{A}) \geq u_i(\check{A}')$. It therefore must be the case that $E^+(\check{A}) \subseteq \cup_{i \in G} E_i^{\check{A}'}$.

   Since $\check{A}'$ enables $E^+(\check{A}')$ (by definition, $\forall I \in E^+(\check{A}) I \in f_{\text{ollow}}(\check{A})$), $E^+(\check{A}') \in \cup_{i \in G} E_i^{\check{A}'}$, and therefore $(E^+(\check{A}') \cup \check{A}) \in \mathcal{A}'^{\check{k}+1}$.

The number of steps in our search is bounded by $O(\max_{A \in SW(\mathcal{A})} l(P(s_0 \rightsquigarrow A)))$. At any step of the procedure, each set of aggregated constraints is either being pruned or extended. The extension will be by at least one constraint that belongs to f_{ollow} of each extended set. Eventually, the furthest set of maximal social welfare will be found. Any extensions of that furthest state will be pruned at the next step. Therefore, the process cannot proceed longer than the number of constraints needed to achieve this set. □

5.7 Using the Procedure on the Example

We now use the iterative procedure to solve the first problem presented in Section 5.2. 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:
$E_1 = [Clr(2), Clr(4)](= E_1^1) \cup [Clr(4), At(2,b)](= E_1^2) \cup [At(4,c), At(2,b)](= E_1^3)$[18]

[18] a_1's plan might be $\langle Move(2,1), Move(4,3)\rangle$. The first operation is enabled since the constraint $Clr(2)$ is satisfied. The $Clr(2)$ constraint is satisfied by s_0, which is included in E_1^1. $Clr(4)$ is needed for a future

E_2 = [$\texttt{Clr}(2)$, $\texttt{Clr}(4)$]$\cup$ [$\texttt{Clr}(4)$,$\texttt{Clr}(2)$, $Clr(5)$]$\cup$ [$Clr(2)$, $Clr(5)$, $On(2,4)$] $\cup$ [$On(2,4)$, $On(5,2)$]
E_3 = [$\texttt{Clr}(2)$]$\cup$ [$\texttt{Clr}(2)$, $Clr(3)$]$\cup$ [$Clr(2)$, $Clr(3)$, $\texttt{At}(3,\bar{c})$]$\cup$ [$\texttt{Clr}(2)$, $Clr(3)$, $At(2,c)$]$\cup$ [$At(2,c)$, $On(3,2)$]

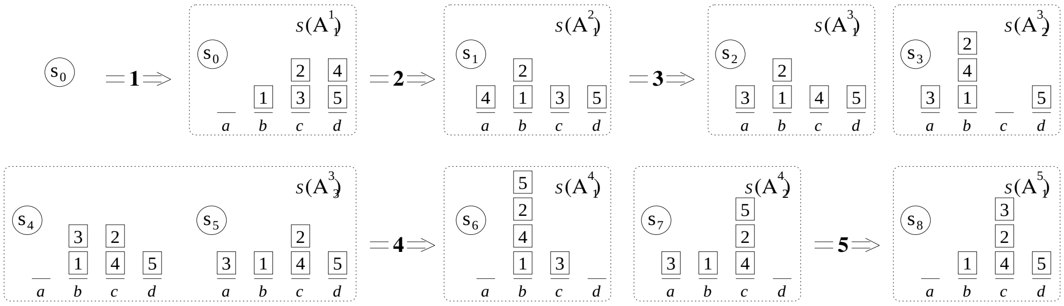

Figure 10: Induced States of the Five Steps

Figure 10 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: $A_1^1 = \{[Clr(2), Clr(4)]\}$. $s(A_1^1)$ 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 $\langle Move(4,\bar{c}), Move(2,4), Move(5,2)\rangle$ 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 $F_{\text{ollow}}(A_1^1)$ for each i). Since all these constraints coexist consistently, $\mathcal{A}^2$ =[$Clr(2)$,$Clr(4)$,$Clr(5)$,$Clr(3)$,$At(2,b)$]. This set induces the single state s_1 (= $s(A_1^2)$) as described in Figure 10. 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.

At the third step, the new added constraints generate 3 possible maximally consistent extensions: A_1^3 =[$At(2,b)$, $At(4,c)$, $Clr(2)$, $Clr(5)$, $Clr(3)$, $At(3,\bar{c})$] inducing s_2,
A_2^3 =[$On(2,4)$, $At(2,b)$,$Clr(2)$, $Clr(5)$, $Clr(3)$, $At(3,\bar{c})$] inducing s_3, and
A_3^3 =[$On(2,4)$, $At(4,c)$, $Clr(2)$, $Clr(5)$,$Clr(3)$, $At(3,\bar{c})$] 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 $\{At(2,b), At(4,c)\}$ (which is in $F_{\text{ollow}}(A_1^3)$), a_2 hands in $\{On(2,4), On(5,2)\}$ and a_3 $\{Clr(3)$, $Clr(2)$, $At(2,c)\}$ (both in $F_{\text{ollow}}(A_3^3)$). 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

operator, but it is also satisfied by s_0, and therefore it too is included in E_1^1. $At(2,b)$ 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.

-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.

5.8 Heuristic Pruning of the Search

Although superior to the central process, the dynamic search process is very complex. Our primary concern has been the finding of *all* states that may be in consensus, rather than the complexity of finding all of them. However, if we relax this demand, and are satisfied with only one state, or with states that are "close" to the actual consensus states, the complexity of the process may be reduced significantly.

One way is to guide the search by using the actual A^* algorithm. With this approach, the evaluation of each alternative state will be as before. However, at each iterative step of extension in the algorithm (Step 2) we choose for further extension only the most promising state according to the evaluation function ($f' = g + h'$). Note that if the heuristic evaluation is to be accurate, then a consensus state will be found after at most $O(\max_{A \in SW(\mathcal{A})} l(P(s_0 \rightsquigarrow A)))$ steps, where at each step only one set of aggregated constraints is extended. However, since sub-plans will tend to interfere with one another, we must search further so as to overcome the misjudgment of the heuristic function, or be satisfied with the first state reached, which may be sub-optimal.

Another computationally expensive component of each step is the generation of the actual optimal plan that derives each intermediate set (A). Here again, we may relax the demand for optimality and consider the plan that is induced by the aggregated set to be the optimal one. Following this approach in the example shown in Section 5.7, the cost of deriving the final consensus state would become 21 instead of 15 (7 $Move$ operators instead of just 5).

Another way of reducing complexity in determining the optimal plan with each intermediate set is to exploit the computational power of participating agents. With this approach, each agent is assigned a subgoal that corresponds to his "contribution" to the set in question A^i (such that $\wedge_i A^i \models A$). The agents then follow a variant of the main algorithm so as to derive the plan that generates the set. In the example, the division into subgoals of the final state would become: $A^1 = \{At(4,c)\}$, $A^2 = \{At(2,c), On(2,4), Clr(5)\}$, $A^3 = \{At(3,c), On(3,2)\}$. Note that in this case, there is no need to consider semi-consistent sets of aggregated constraints since all subgoals are coherent. For that same reason, the heuristic evaluation is much more accurate.

Using this approach, the actual multi-agent plan may be constructed during the process itself. The construction may be made at Step 2 of the algorithm. Each agent bids for each action that each extension implies. The bid is based on the sequence of i's previous actions in the extended set. Thus, the minimal cost sequence may be determined.

Yet another way to prune the search process is to define a dominance relation over alternative extensions at each step, and ignore dominated extensions. An example might be to prefer an extension that satisfies in full the declaration of one agent, over one that only partially satisfies two different declarations. Or, given an aggregated set A, we may prefer its extension E that induces a grounded constraint I over another extension $\bar{E}$ that induces a grounded constraint $\bar{I}$ (such that $I \wedge \bar{I} \models F_alse$) if $I \in A$ while $\bar{I} \notin A$ or if $\bar{I}$ appears in some agent's final goal, while I does not (intuitively, we can in this way avoid directions in which constraints are added that are merely stumbling blocks). Following this heuristic in Example 5.7, states s_3 and s_6 would not be considered since they will be dominated by states s_2 and s_7 respectively. In many cases $\bar{E} \setminus \bar{I}$ would be preferable to $\bar{E}$ itself. Following Example 5.7 again, s_1 will contain $\{On(4,5)\}$ instead of $\{On(4,a), Clr(5)\}$ and the "premature" removal of block 4 will be avoided.

6 Specification and Aggregation of Constraints

There are several important aspects of, and requirements for, the procedure above:

- The procedure's success depends on each agent identifying and declaring only the absolutely necessary constraints needed for its individual plan to succeed.
- It is necessary that *temporary* conflicts between the agents' plans not cause deadlocks (that is, if there is a temporal order that can later resolve them). This phenomenon is achieved in our procedure by allowing the existence of semi-consistent sets of constraints and then having agents express worth preferences over the states that they induce.
- The generation of consensus sets of constraints is based on the aggregation of individual sets of constraints. This is not always trivial to do.

Figure 11 shows three simple scenarios in the slotted blocks world that illustrate the above issues. In these examples, two agents can achieve a consensus state that fully satisfies both agents' goals. There are three slots, and some blocks in each world. Only the $Move$ operator is available and it costs 2.

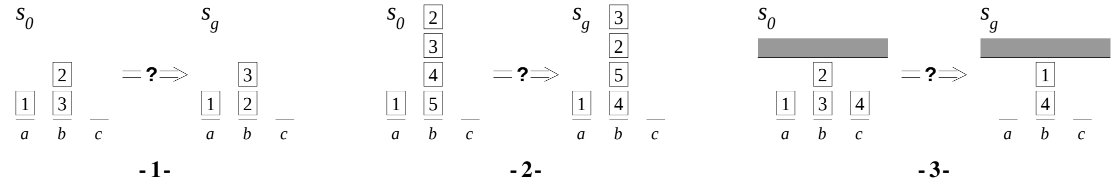

Figure 11: Three Scenarios in the Slotted Blocks World

6.1 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 11. Two agents want to achieve the following goals: $g_1 = \{At(1,a), Clr(1)\}$ and $g_2 = \{On(3,2), At(2,b)\}$. 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 follow $\langle Move(2,x), Move(3,\bar{2}), Move(2,b), Move(3,b)\rangle$.[19] 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. Although neither of these operations has to do with a_2's final goal, by including either of these alternatives as one of its constraints, a_2 would encounter a_1's opposition, thus (perhaps) preventing his own goal from being achieved.

However, by realizing that the purpose of the second *Move* operator is just to achieve $At(3,\bar{b})$, the danger of conflict may be avoided. Thus, if a_2 declares his set of constraints to be $[Clr(2)] \cup [Clr(2), Clr(3)] \cup [Clr(2), Clr(3), At(3,\bar{b})] \cup [Clr(2), Clr(3), At(2,b)] \cup [At(2,b), On(3,2)]$, no conflict arises. At any step, a_1's declaration equals his full goal's description. Therefore, at the third step of the procedure, the aggregated set of constraints would become $[Clr(2), Clr(3), At(3,\bar{b}), Clr(1), At(1,a)]$, which is fully satisfied by the state $\{At(1,a), At(2,b), At(3,c)\}$. This state is one step distant from the goal state. The described goal state will thus be generated at the next step. Note that $At(2,\bar{b})$, which a_2's plan achieves after the first step, is not required for further progress, and therefore should not be included in the set of constraints at all. Conflict was avoided because the duration of the temporary conflict between the two agents' plans was short enough to be resolved by invoking one more operator.

6.2 Aggregation of Temporary Constraints

Unfortunately, this is not always the case. In the second scenario, a_1's goal is the same, but a_2's goal is now $\{On(5,4), On(3,2), At(2,b), At(4,b)\}$. 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 hide the conflict. Here, for example, $At(2,\bar{b})$, 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 *temporary* conflicts between the agents' plans not cause deadlocks (that is, if there is a temporal order that can later resolve them). This phenomenon 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 (I_2) would then

[19] $\bar{y}$ denotes any location excluding y's.

become:[20] $[\mathtt{Clr(2)}] \cup [\mathtt{Clr(2)}, \mathtt{At(2,\bar{b})}] \cup [\mathtt{At(2,\bar{b})}, \mathtt{Clr(4)}, \mathtt{At(3,\bar{2}\bar{b})}] \cup [\mathtt{At(2,\bar{b})}, \mathtt{Clr(4)}, \mathtt{At(3,\bar{2}\bar{b})}, \mathtt{Clr(5)}, \mathtt{At(4,\bar{b})}] \cup [\mathtt{At(2,\bar{b})}, \mathtt{Clr(4)}, \mathtt{At(3,\bar{2}\bar{b})}, \mathtt{Clr(5)}, \mathtt{At(5,\bar{4}\bar{b})}] \cup [\mathtt{At(2,\bar{b})}, \mathtt{Clr(4)}, \mathtt{At(3,\bar{2}\bar{b})}\ \mathtt{Clr(5)}, \mathtt{At(5,\bar{4}\bar{b})}, \mathtt{Clr(2)}, At(4,b)] \cup [\mathtt{At(3,\bar{2}\bar{b})}, Clr(2), At(4,b), On(5,4), Clr(3)] \cup [Clr(2), At(4,b), On(5,4), Clr(3), At(2,b)] \cup [At(4,b), On(5,4), On(3,2), At(2,b)]$

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. Since all first eight sets of combined constraints (in the first eight steps) are semi-consistent, they are taken into account and voted on. In addition, since all these sets are of increasing value (a_1 is always satisfied with his goal not being violated, and a_2 gets closer to his own goal), the sets are not pruned. Thus, the process continues, leading in the ninth step to the (real) mutual goal state.

6.3 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 11 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 $At(1,b)$ while $g_2 = At(4,b)$. We assume that the function $fs(b)$ returns the available free space (in number of blocks) left at slot b. As in any scenario, s_0 satisfies I_i^1 of both agents.[21] Following the second step of their plans, each of the agents has as temporary constraints $\mathtt{fs(b)} \geq \mathtt{1}$ and $Clr(x)$, where x is the block each 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 $fs(x) \geq 1$ will be $fs(x) \geq 2$, or simply $fs(x) \geq 1$. Had block 4 been located on block 1 in the initial state of our example, the first solution would be appropriate. Following it in the given scenario, however, would yield the aggregated set of constraints in the second step to be $[Clr(1), Clr(4), fs(b) \geq 2]$. 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 free space should stay 1, leading to a state which is only three *move* steps distant from the initial state. Unfortunately, it not clear how in general this subtle analysis is to be done (see [2] for further discussion on the difficulty of handling functional constraints).

6.4 Global Worth Evaluation of Sets of Constraints

The process also counts on the fact that the agents' worth functions increase monotonically with the number of satisfied constraints (within the maximal gap bound δ). This requirement yields the non-decreasing value of the extended set of constraints. Thus, the pruning

[20] We use typewriter style to denote temporary constraints.

[21] $I_1^1 = [Clr(1), Clr(2), \mathtt{fs(c)} \geq \mathtt{1}]$ and $I_2^1 = [Clr(4), Clr(2), \mathtt{fs(a)} \geq \mathtt{1}]$. Note that by its nature the free space constraint is temporary, since it always is a precondition of an action that violates it!

of unpromising states is enabled. However, the fact that the individual worth functions monotonically increase may sometimes lead to counter-intuitive results. Consider again the last example. Each agent can achieve his own goal by two $Move$ steps, whereas to achieve both goals it takes five. Therefore, as long as the cost of a multi-agent $Move$ is not less than $\frac{4}{5}$ of a single agent $Move$, s_0 will be the optimal state using our social welfare criterion.

7 Self-interested Rational Agents

Throughout this paper we have assumed that agents will benevolently follow the rules set forth in our procedures. However, if agents are self-interested and rational (utility maximizers), they need to be motivated to contribute to the global plan and follow the rules of the process.

One way to motivate rational agents to work is to have, built into the system, some kind of incentive or reward associated with this work [28, 30, 57, 53]. Without loss of generality, we will call such an incentive a *payment*. A straightforward payment policy would be to pay each agent the cost it associates with its performed work (or contribution). Thus, in the procedures suggested above, each agent should be paid for carrying out an operator, according to the bid it gives for that operator.

Unfortunately, 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 cost values, in an attempt to gain higher payment. On the other hand, in constructing an optimal plan, it is critical that agents, at each step, express their true cost values (bids). 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 [12, 13, 14].

Moreover, even if we employ an incentive compatible bidding mechanism (i.e., such that agents will find true bidding to be in their best interest) our problems are not solved. A fundamental part of each of the merging processes that were introduced above is the heuristic evaluation (or utility) that each agent assigns to each alternative extension. Here again rational agents may find it tempting to manipulate their vote so as to bend the global decision in their favor. We need the agents to be honest both in their declaration of constraints during the merging process, and also while bidding during that process.

In the two following subsections we suggest a solution to these two problems. Section 7.1 considers a simple mechanism that induces true bids over operators, and Section 7.2 considers a mechanism that induces true estimates of extensions.

7.1 Deriving True Bids over Induced Operators

The first potential problem is that agents may declare dishonest bids over the induced operator of each extension. By doing so, they may prevent the construction of the actual optimal multi-agent plan. Fortunately, there do exist solutions to this problem, such that the above planning mechanisms can be used even when the agents are not necessarily benevolent and honest. By using a variant of the Vickrey Mechanism [50] (more commonly known as the "second-best auction") it is possible to ensure that all agents will bid honestly and will declare their constraints honestly. This is done through minor changes to the bidding phase (Step 4, Step 5 and Step 7 that were respectively described in Section 3.4, Section 4.5 and Section 5.6).

First, the agents' bids will be submitted sealed (the individual bids will be publically revealed only after all agents have submitted their bids). Second, as before, an operator will be assigned to the agent that declares the minimal cost (bid); however, the actual payment will be equal to the *next lowest bid.* Note that under this scheme total payment for the plan will be greater than its actual cost.

A more formal analysis of the process supports the following claim: at *any* bidding step t of each of the procedures above, i's best strategy is to bid over the induced actions at that step (all operators in $P(Ex^*(A^{t+}))$ according to his true cost.

To understand why this is the case, consider a_i facing a bid over Op_k. Assume that the real cost of performing the operator is $X = c_i(Op_k)$. To "win" the operator, a_i might be tempted to bid $X - \delta$, counting on the second-best bid to be $\geq X$. However, it might be the case that the second-best bid is $X - \epsilon$ (such that $\epsilon < \delta$) and a_i would win the bid but lose ϵ. (If the second best bid is $\geq X$ then bidding X would be as good as bidding $X - \delta$, so there is no reason to take the risk.) On the other hand, a_i might be tempted to bid $X + \delta$ in order to be paid more. But doing so, he risks losing the bid altogether; another agent may win with a bid of $X + \epsilon$ and a_i would not be paid at all (while by bidding honestly he would be paid ϵ). If the winning bid will be $X + \delta$, then a_i would gain no advantage since his payment will be determined by the second-best bid (and he might as well bid X).

7.2 Deriving True Estimates of Sets of Constraints

The second potential problem is that agents may declare dishonest heuristic values over the extensions themselves. As an example, consider an agent who is significantly superior to all other agents at rotating blocks. Employing the above bidding scheme, such an agent may find it rational to bend the search path for the optimal multi-agent plan in directions that would induce more rotate operators. The agent does so, trusting the bidding mechanism to ultimately assign him these operators, and with a payment much higher than the agent's true cost (the second-best bid).

Fortunately, there do exist solutions to this potential problem as well, such that the above plan choice mechanisms can be used even when the agents are not necessarily benevolent and honest. This can be done by minor changes to the procedures of Sections 3.4, 4.5 and 5.6, that allow it to use a variant of the Clarke Tax mechanism (CTm).

In [12] 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.

We here describe how the CTm can be integrated into our techniques, to derive honest votes in each of the evaluation steps of the above procedures.

To be integrated into the above planning procedures, the CTm itself needs to be adjusted. In our procedures, 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 $\sum_{i=1}^n d_i^k(A^k)$.

- 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 employment of the mechanism also requires several changes to the procedures given above:

1. Bids should be sealed (or given simultaneously).

2. The global evaluation function (what was considered to be the social welfare function, f^U, in Section 5.3 and h' in the first two algorithms) must be taken to be the (weighted) sum f_{sum}^U (or average) of the individual utilities, so as to ensure that all agents will vote honestly.

3. 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. For example, an agent voting over A_0 and A_1 might give the first a value of 0 and the second a value

of 5. If that agent subsequently voted over A_2 and A_3, he might assign the first 0 and the second 8—but if he had voted over all four alternatives at once, A_2 and A_3 would have been shifted up by 3, giving votes of 0, 5, 3, and 11 to the candidates.

Therefore, we 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 set A in the vote is exactly $f^U_{sum}(A)$.

Some specific adjustments should also be made to several of the algorithms. In the first two algorithms (Sections 3.4 and 4.5), the global evaluation is already taken to be the sum of individual estimates (h'_i). However, we can no longer assume that this estimate will necessarily be the additional estimated cost according to the individual sets of constraints. This is a crucial assumption for the efficiency of these two procedures. We therefore have to employ a different policy that will make this estimate the true evaluation of each agent. (Notice that in the third procedure we still assume *any* individual worth functions $[u_i]$.)

In some sense, this new policy is aimed at transforming the two first scenarios so that they resemble the third scenario. First, instead of being paid for each performed operator, each assigned subgoal (or task) will have an associated payment, to be given to the assigned agent upon its completion. Second, instead of voting (declaring h'_i) only over alternative extensions (Step 5 of the two first algorithms), the agents will vote over each extension and all possible work distributions that it induces. That is, instead of being considered as one candidate alternative, each extension will be represented as several alternatives. Third, in the second scenario, if its task has been changed, an agent will be paid for the work he performed prior to that change.

Under this payment policy, the true worth (or utility) that each agent will assign to each alternative extension will become exactly the distance (in terms of additional cost) to its assigned goal minus his share of the work. Since truth telling becomes the dominant strategy (due to the CTm), we are guaranteed to have the correct estimation (within δ) of each alternative.

The third planning algorithm (Section 5.6) should also be updated; 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, knowing that this vote would not be subject to the Clarke Tax. 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 end of the process, each agent is fined the Clarke Tax with respect to the final group choice.

Theorem 4 *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 *normalized preferences $\vec{v_i^k}$ (that is, subtracting $w_i(s_0)$ from $w_i(A)$ for any A).*

Proof:

To show that declaring $\vec{v_i^k}$ is the dominant strategy, we have to show that agent a_i's utility from declaring it is greater than any other declaration $\vec{d_i^k}$:

$u_i^k(f(D_{\neg i}^k, \vec{v_i^k})) - u_i^k(f(D_{\neg i}^k, \vec{d_i^k})) =$

Expanding the utilities u into worth minus tax, we get:
$w_i(f(D_{\neg i}^k, \vec{v_i^k})) - t_i^k(f(D_{\neg i}^k, \vec{v_i^k})) - w_i(f(D_{\neg i}^k, \vec{d_i^k})) + t_i^k(f(D_{\neg i}^k, \vec{d_i^k})) =$

Expanding the tax t into its components, we get:
$w_i(f(D_{\neg i}^k, \vec{v_i^k})) - \sum_{j \neq i} d_j^k(f(D_{\neg i}^k)) + \sum_{j \neq i} d_j^k(f(D_{\neg i}^k, \vec{v_i^k}))$
$-w_i(f(D_{\neg i}^k, \vec{d_i^k})) + \sum_{j \neq i} d_j^k(f(D_{\neg i}^k)) - \sum_{j \neq i} d_j^k(f(D_{\neg i}^k, \vec{d_i^k})) =$

Eliminating equivalent plus and minus terms and adding and subtracting $w_i(s_0)$, we get:
$w_i(f(D_{\neg i}^k, \vec{v_i^k})) - w_i(s_0) + \sum_{j \neq i} d_j^k(f(D_{\neg i}^k, \vec{v_i^k})) - [w_i(f(D_{\neg i}^k, \vec{d_i^k})) - w_i(s_0) + \sum_{j \neq i} d_j^k(f(D_{\neg i}^k, \vec{d_i^k}))] =$

By v_i's definition:
$v_i(f(D_{\neg i}^k, \vec{v_i^k})) + \sum_{j \neq i} d_j^k(f(D_{\neg i}^k, \vec{v_i^k})) - [v_i(f(D_{\neg i}^k, \vec{d_i^k})) + \sum_{j \neq i} d_j^k(f(D_{\neg i}^k, \vec{d_i^k}))] =$

Let $\hat{A}^k$ be the set that maximizes $v_i^k(A^k) + \sum_{j \neq i} d_j^k(A)$ (i.e., $\hat{A}^k = f(D_{\neg i}^k, \vec{v_i^k})$), and let $\hat{k}^k$ be the set that maximizes $\sum_{i=1}^n d_i^k(A^k)$ (i.e., $\hat{k}^k = f(D_{\neg i}^k, \vec{d_i^k})$). Then (by the definition of the choice function f) we get:

$v_i(\hat{A}^k) + \sum_{j \neq i} d_j^k(\hat{A}^k) - [v_i(\hat{k}^k) + \sum_{j \neq i} d_j^k(\hat{k}^k)] =$

By $\hat{A}^k$'s definition:
$max_{k, A^k}[v_i(A^k) + \sum_{j \neq i} d_j^k(A)] - [v_i(\hat{k}^k) + \sum_{j \neq i} d_j^k(\hat{k}^k)] \geq 0$ □

The theorem above demonstrates a subtle consequence of the "independence of irrelevant alternatives" property of the original CTm. An agent cannot change the winning alternative simply by causing other alternatives to be pruned. Moreover, directly causing another alternative to win (either by overbidding that alternative, or underbidding the rightful winner) will expose the agent to the Clarke Tax.

Table 12 describes the resulting accumulated iterative Clarke Tax vote in Example 5.7. Each row of the table shows several pieces of information regarding an agent. First, his preferences for each state are listed. Then, the total score that each state would have received, had the agent *not* voted, are listed. An asterisk marks the winning choice in each situation.

	Worth of each state								Score of each state without i								Tax
	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	
a_1	2	5	-7	0	0	-11	-5	-5	4	6	10	$*12$	$*12$	6	10	10	0
a_2	2	1	5	4	4	9	7	-1	4	$*10$	-2	8	8	-14	-2	6	2
a_3	2	5	5	8	8	-3	3	11	4	$*6$	-2	4	4	-2	2	-6	2
sum	6	11	3	$*12$	$*12$	-5	5	5									4

Figure 12: Cumulative Voting using the Clarke Tax Mechanism

As can be seen from the table, 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.

8 Related Work

The computational benefits of working separately with subgoals have long been recognized. In this section we briefly address work that has aimed at exploiting these benefits and is most related to our own research. In general, our approach differs from previous work in that it combines the following attributes:

- The search is directed by well-understood AI search methods that use, as the basis for the heuristic function, a qualitative measure of the global plan that is formed. Therefore, compared with other planning algorithms, the process is likely to find plans of higher quality. There are two reasons for this improved quality. First, the heuristic function that we use is not purely local, as in other planners; instead it makes use of fully-expanded single-agent plans for the heuristic measure. Second, the heuristic measure itself is based on cost, and not just on the length of the plan (as in virtually all work in Planning, e.g., [34]).

  Although in the first two scenarios below we consider only the cost of the plan, and it is only in the third scenario that we address its global utility, see [11] for possible ways to expand our approach to deal with utilities in general.

- The entire process has the following advantages from a complexity point of view. First, the branching factor of the search space is constrained by the individual plans' constraints. Second, the variants of the A^* algorithm use a relatively good heuristic function, because it is derived "bottom-up" from the plans that the agents have already generated (not simply an artificial h' function). Third, generation of successors in the search tree is split up among the agents (each doing a part of the search, in parallel, for a successor). Fourth, the heuristic function is calculated only for maximally "feasible" alternatives (infeasible alternatives need not be considered).

- Individual plans are represented by sets of ordered constraints. We use ordered propositions instead of sequences of operators for the following reasons. First, the plans for individual agents that are generated serve only to *guide* the heuristic search for the actual global multi-agent plan. The *actual* multi-agent action that establishes a proposition is determined only during the merging process itself. This is essential for the efficiency of the resulting multi-agent plan.[22] Second, the propositions encode all the information needed for the heuristic evaluation. Third, by dealing with propositions we achieve more flexibility (least commitment) in the merging process, both in the choice of operators and in their bindings.

- The search is done through the space of states. Thus, the essential search method is similar to the search employed by progressive world-state planners. Many researchers consider a search through the space of states to be inferior to a search, as conducted by POCL planners (e.g., [2, 34, 40]), through the space of plans [36]. The reason is that any (nondeterministic) choice of action within the first method also enforces the timing of that action (and thus, a greater breadth of search is needed to ensure completeness, e.g., in the Sussman anomaly). However, if there would actually exist an oracle that could perform the right nondeterministic choice (in current implementations of POCL planners, this choice is implemented by backtracking in a brute force search), there is no reason to prefer one search method over the other.[23] In our approach, the heuristic evaluation guides the choice, and thus improves the performance of the state-space search approach (see [49] for other examples in which state-space search methods outperform POCL planners). And indeed, given the individual sub-plans, our planning procedure need consider only a small number of feasible expansions (relative to the full search space), among which the heuristic evaluation "foresees" most commitments that may result in backtracking. Thus, it also becomes possible and worthwhile to avoid

[22] For example, it might be the case that towards the achievement of his assigned subgoal, a_i planned to perform op_k in order to establish proposition P, but in the multi-agent plan P will actually be established by a_j performing op_r. Therefore, what counts for the global multi-agent plan is *what* is established, rather than *how* it is established. Note, however, that each declared proposition is guaranteed to be satisfied, by the original agent's operator if not by someone else's.

[23] Actually, if a plan is represented as a totally-ordered sequence of actions, and we consider the path in the first method as a plan, the entire search may be viewed as a search in the plan-space and vice versa. Thus the distinction between the two search approaches is more a matter of point of view.

the causal-link-protection step of the POCL planners.

- Positive and negative interactions among separately generated plans (the correspondence of positive and negative threats within a single POCL plan) are handled within the same process.
- The parallelism of the actual execution of the multi-agent plan is determined as the plan is constructed.
- Based on the concept of partial satisfaction, the technique presents a new concept of solution for conflicting individual goals and guides the construction of the plan or plans that derive this solution. In contrast to the typical binary attitude towards goal achievement (either the plan succeeded or failed), our technique constructs a plan that results in a compromise outcome that partially satisfies the multiple agents involved.
- The technique allows us to deal with non-benevolent agents, and prevent them from manipulating outcomes. The traditional assumption that each agent will contribute to the process of constructing the global plan regardless of the individual effort that requires, may be replaced with the (perhaps more domain-appropriate) assumption that agents are rational.

An approach similar to our own is taken in [38, 55] 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. The basic idea there is to try and construct a global plan by repeatedly (branch and bound) merging complete plans that achieve the separate subgoals. Due to the complexity of the general problem, certain restrictions are imposed on the types of interactions allowed. In our approach, there are no prior restrictions, the global plan is created incrementally out of fractions of sub-plans, and agents do the merging in parallel.

Other research that makes use of separately planning for subgoals in order to achieve one common goal is described in [22]. Instead of working with precise sub-plans, the author introduces the concept of “approximate” plans (i.e., plans that under most circumstances will achieve their goals). Given several sets, each with a sufficient number of approximate sub-plans, an approximate global plan may be determined by their intersection. The notion of approximate sub-plans is aimed somewhat at achieving what our approach achieves by representing sub-plans only as sets of absolutely necessary constraints. But, essentially, as with the work mentioned above, this method also treats sequences of actions as atomic units, and then merges them into larger structures made up of these units.

In [19] it is shown how to handle positive interactions efficiently among different parts of one given global plan. The merging process looks for redundant operators within the same *grounded linear plan* in a dynamic fashion (as opposed to aggregating constraints of several separately generated plans). An assumption is made that knowledge is available about what operators can be merged, and for each set of these operators, what the merged operators are. To achieve an optimal final plan, it takes that algorithm $O(\prod_{i=1}^{n} l(P(g_i)))$ steps (an approximation algorithm that is presented there takes polynomial time), while given the assumptions that are being made in that work, our algorithm would find the global plan within $O(\max_i l(P(g_i)))$ steps.

In [54], on the other hand, it is shown how to handle conflicts efficiently among different parts of a given plan. Conflicts are resolved by transforming the planning search space into a constraint satisfaction problem. The transformation and resolution of conflicts is done using a backtracking algorithm that takes cubic time. In our approach, both positive and negative interactions are addressed simultaneously. Positive interactions are detected during each aggregation step of the merging process. The most efficient merge is determined by the heuristic value that each extension scores. Similarly, our methods for conflict resolution are employed within a single aggregation step: promotion of clobberer, demotion of clobberer, and separation [2] are all considered when a possible extension is generated. Among these possibilities, the optimal one (according to the heuristic function) is chosen. Introduction of a white knight, if necessary, is detected at the next step of aggregation. This phenomenon of ours is enabled by the strong assumption that sub-plans are represented as abstract sets of necessary constraints and that each aggregated extension has an associated realistic heuristic value. The fact that aggregation is done linearly does away with the need for the complete look-ahead that is assumed by others' methods. This fact is essential to interleaving the process of planning with execution.

Our approach also resembles the GEMPLAN planning system [33, 32]. There, the search space 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. The example given there, where regions are determined with respect to different expert agents that share the work of achieving a global goal, can also be addressed in a natural way by the algorithm presented here (although we assume that all the domain's resources can be used by the participating agents). The GEMPLAN model served as a basis for the DCONSA system [41] 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 planning technique that we suggest relates also to the approach suggested in [51]. There too the planning process is viewed as the process of incremental constraint posting. A method is suggested for assigning preferences to sets of constraints that will direct the planner. There is also the use of partial goal satisfaction (see [52] as well). However, the evaluation and comparison between alternatives is done according to the global view of the single planner, and is based on pre-defined dominance relations.

The concept of solution that our algorithm employs (maximization of social welfare) also resembles the approach taken in CONSENSUS [3] where several expert systems "elect" a plan that scores the highest rating with respect to the individual points of view. There, however, the candidates in the election are alternative *complete global* plans that each expert generates by itself. Agents (expert systems) do not share in the construction of the global plan (neither computationally, nor by sharing perspectives), but rather choose among plans that were constructed individually.

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 [37], 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 7.2), our approach allows us to deal with a rational selfish agent (there is no need to assume that the agents are benevolent).

9 Conclusions

In this paper, we presented a novel multi-agent planning approach. The procedure relies on an *a priori* division of the global goal into subgoals. Agents solve local subgoals, and then merge them into a global plan. By making use of the computational power of multiple agents working in parallel, the process is able to reduce the total elapsed time for planning as compared to a central planner. The optimality of the procedure is dependent on several heuristic aspects, but in general increased effort on the part of the planners can result in superior global plans. The overall technique can be used in a way that results in an anytime planning procedure.

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.

The techniques presented here are also applicable in any situation where multiple agents are attempting to merge their individual plans or resolve inter-agent conflict when execution time is critical.

Acknowledgments

We would like to thank Martha Pollack for helpful discussions, and the anonymous reviewers for their comments that contributed to the improvement of this paper.

This work has been supported in part by the Air Force Office of Scientific Research (Contract F49620-92-J-0422), by the Rome Laboratory (RL) of the Air Force Material Command

and the Defense Advanced Research Projects Agency (Contract F30602-93-C-0038), by an NSF Young Investigator's Award (IRI-9258392) to Prof. Martha Pollack, and by the Israeli Ministry of Science and Technology (Grant 032-8284) and the Israel Science Foundation (Grant 032-7517).

References

[1] M. Boddy and T. Dean. Solving time-dependent planning problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 979–984, Detroit, Michigan, August 1989.

[2] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–377, July 1987.

[3] R. Clark, C. Grossner, and T. Radhakrishnan. CONSENSUS: 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.

[4] D. 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.

[5] D. D. Corkill. *A Framework for Organizational Self-Design in Distributed Problem Solving Networks*. PhD thesis, University of Massachusetts, Amherst, MA, December 1982.

[6] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 49–54, Saint Paul, Minnesota, August 1988.

[7] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 49–54, St. Paul, Minnesota, August 1988.

[8] J. Doyle and M. P. Wellman. Impediments to universal preference-based default theories. *Artificial Intelligence*, 49(1–3), 1992.

[9] E. H. Durfee. A cooperative approach to planning for real-time control. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 277–283, San Diego, California, November 1990.

[10] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Cooperation through communication in a distributed problem solving network. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, chapter 2, pages 29–58. Morgan Kaufmann Publishers, Inc., Los Altos, California, 1987.

[11] E. Ephrati, M. E. Pollack, and J. S. Rosenschein. Exploitation of decision theory techniques in multi-agent planning (extended abstract). In *Working notes of the AAAI Spring Symposium on Decision-Theoretic Planning*, pages 90–98, Stanford, CA, March 1994.

[12] 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.

[13] 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.

[14] E. Ephrati and J. S. Rosenschein. Distributed consensus mechanisms for self-interested heterogeneous agents. In *Proceedings of the First International Conference on Intelligent and Cooperative Information Systems*, pages 71–79, Rotterdam, The Netherlands, May 1993.

[15] E. Ephrati and J. S. Rosenschein. Multi-agent planning as a dynamic search for social consensus. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 423–429, Chambery, France, August 1993.

[16] E. Ephrati and J. S. Rosenschein. Planning to please: Following another agent's intended plan. *Journal of Group Decision and Negotiation*, 2(3):219–235, 1993.

[17] E. Ephrati and J. S. Rosenschein. Divide and conquer in multi-agent planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 375–380, Seattle, Washington, July 1994.

[18] J. J. Finger. *Exploiting Constraints in Design Synthesis.* PhD thesis, Stanford University, Stanford, CA, 1986.

[19] D. E. Foulser, M. Li, and Q. Yang. Theory and algorithms for plan merging. *Artificial Intelligence*, 57(2-3):143–181, October 1992.

[20] M. Georgeff. Communication and interaction in multi-agent planning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 125–129, Washington, D.C., August 1983.

[21] 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.

[22] M. L. Ginsberg. Approximate planning (extended abstract). In *Proceedings of the Second International Conference on AI Planning Systems*, pages 68–73, Chicago, Illinois, June 1994.

[23] P. Haddawy and S. Hanks. Issues in decision-theoretic planning: Symbolic goals and numeric utilities. In *Proceedings of the 1990 DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control*, pages 48–58, November 1990.

[24] S. Hanks and R. J. Firby. Issues and architectures for planning and execution. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 59–70, San Diego, California, November 1990.

[25] K. Kanazawa and T. Dean. A model for projection and action. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 985–990, 1989.

[26] M. J. Katz and J. S. Rosenschein. Verifying plans for multiple agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 5:39–56, 1993.

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

[28] S. Kraus. Agents contracting tasks in non-collaborative environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 243–248, 1993.

[29] S. Kraus, M. Nirkhe, and D. Perlis. Deadline-coupled real-time planning. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 100–108, San Diego, California, November 1990.

[30] S. Kraus, J. Wilkenfeld, and G. Zlotkin. Multiagent negotiation under time constraints. Computer Science Technical Report Series CS-TR-2975, University of Maryland, College Park, Maryland, October 1992.

[31] B. Krulwich. Planning for soft goals. In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems*, pages 289–292, College Park, Maryland, November 1992.

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

[33] A. L. Lansky and D. S. Fogelsong. Localized representation and planning methods for parallel domains. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 240–245, Seattle, Washington, July 1987.

[34] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 634–639, Anaheim, California, July 1991.

[35] D. McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[36] S. Minton, J. Bresina, and M. Drummond. Commitment strategies in planning: A comparative analysis. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 259–265, August 1991.

[37] 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, California, November 1990.

[38] 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.

[39] E. P. D. Pednault. Formulating multiagent dynamic-world problems in the classical planning framework. In Michael P. Georgeff and Amy L. Lansky, editors, *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, pages 47–82, San Matheo, CA, 1987. Morgan Kaufmann.

[40] J. Penberthy and D. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the Third International Conference on Knowledge Representation and Reasoning*, pages 103–114, Cambridge, MA, 1992.

[41] 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.

[42] J. Rawls. *A Theory of Justice*. Belknap, Cambridge, MA, 1971.

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

[44] J. S. Rosenschein and M. R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 91–99, Los Angeles, August 1985.

[45] R. G. Smith. *A Framework for Problem Solving in a Distributed Processing Environment*. PhD thesis, Stanford University, 1978.

[46] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, December 1980.

[47] Christopher J. Stuart. An implementation of a multi-agent plan synchronizer. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 1031–1035, Los Angeles, California, 1985.

[48] G. J. Sussman. *A Computational Model of Skill Acquisition.* American Elsevier, New York, 1975.

[49] M. Veloso and Jim Blythe. Linkability: Examining casual link commitments in partial-order planning. In *Proceedings of the Second International Conference on AI Planning Systems*, pages 170–175, Chicago, Illinois, June 1994.

[50] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[51] M. P. Wellman. Dominance and subsumption in constraint-posting planning. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 884–889, Milan, Italy, August 1987.

[52] M. P. Wellman and J. Doyle. The preferential semantics for goals. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 698–703, Anaheim, California, July 1991.

[53] M. P. Wellman. A general equilibrium approach to distributed transportation planning. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 282–289, San Jose, California, July 1992.

[54] Q. Yang. A theory of conflict resolution in planning. *Artificial Intelligence*, 58(1-3):361–393, December 1992.

[55] Q. Yang, D. S. Nau, and J. Hendler. Merging separately generated plans with restricted interactions. *Computational Intelligence*, 8(2):648–676, February 1992.

[56] S. Zilberstein and S. J. Russell. Efficient resource-bounded reasoning in AT-RALPH. In *Artificial Intelligence Planning Systems: Proceedings of the First International Conference*, pages 260–266, College Park, Maryland, June 1992.

[57] G. Zlotkin and J. S. Rosenschein. A domain theory for task oriented negotiation. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 417–422, Chambery, France, August 1993.