Multi-Agent Planning as the Process of Merging Distributed Sub-plans

Eithan Ephrati Jeffrey S. Rosenschein
Computer Science Department
Hebrew University, Jerusalem, Israel
tantush@cs.huji.ac.il, jeff@cs.huji.ac.il

Abstract

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, given those original subgoals.

1 Introduction

The subject of multi-agent planning has been of continuing concern in Distributed Artificial Intelligence (DAI). Some of the earliest work in the field, such as Smith's Contract Net [14] and Corkill's work on distributing the NOAH planner [2], were directly addressing the question of how to get groups of agents to carry out coordinated, coherent activity. These questions have occupied researchers both in the Cooperative Problem Solving subarea of DAI (researchers concerned with centrally designed groups of agents with a global goal), as well as those working on Multi-Agent Systems (collections of self-interested agents with no global goal). More recent research on the issue of multi-agent planning includes that of Martial [18, 17], Katz and Rosenschein [10], Durfee [5, 4, 6], and Lansky [12].

The term multi-agent planning has been used to describe both "planning *for* multiple agents" (where the planning process itself may be centralized), and "planning *by* multiple agents" (where the planning process is distributed among the agents). In this paper, we are concerned with the latter paradigm: we investigate how a group of agents can cooperatively construct a plan that they will then, as a group, implement.

In this paper, we present a multi-agent planning procedure to achieve a global goal. The procedure relies on a given *a priori* division of the global goal into subgoals. Agents solve their local subgoals, which are then merged 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.

1.1 Exploitation of Subgoals and Sub-plans in Multi-Agent Planning

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 optimal path from the initial state to the goal state). The time complexity of the planning problem is then $O(b^d)$ [11].

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).

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 [11], if the subgoals are independent or serializable,[1] the central multi-agent planning time complexity can be reduced to $\sum_i((n \times b_i)^{d_i})$.

This phenomena 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 $\max_i(n \times b_i)^{d_i}$. The global plan is then constructed out of local plans that are based upon the agents' local knowledge. This approach removes the need for a central planner that has global knowledge of the domain and the agents involved.

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

In this paper we suggest an approach to multi-agent planning that contains heuristic elements. Our method makes use of pre-existing subgoals, and derived sub-plans, to construct 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, given those original subgoals.

2 The Scenario

Our scenario involves a group $A = \{a_1, \ldots, a_n\}$ of n agents. These agents are to achieve a global goal G (we here assume that the agents are benevolent and cooperative, but see Section 6). The global goal, G, has been divided into n subgoals ($\{G_1, \ldots, G_n\}$), 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.

2.1 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 ($\cup_i g_i \models G$).

  The set of constraints (and domain axioms) that specifies the interface between these subgoals is denoted by $D(G)$. We assume that for any 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.

[1] 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.

- The subgoals may be (partially) ordered. With respect to this order and the constraints $D(G)$, we denote the prior state that must precede a subgoal G_i by G_i^0. However, it is not a *requirement* that the subgoals be ordered. It may be possible to carry them out in parallel, interleaved, etc. Usually, in fact, G_i^0 will simply be the initial state, for most i.
- 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 g_i^0.
- Each agent has a *cost function* ($C: OP \times \mathcal{A} \Rightarrow \mathbb{R}$) over the domain's operators [9]. The cost of a_j's plan $c_j(p_j)$ is defined to be $\sum_{k=1}^{m} c_j(op_k)$.

2.2 An Example

Consider a scenario in the discrete grid sticky blocks world.[2] 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 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: **Clear**(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 $r(b)$ and $l(b)$ return the grid region of b's left edge, and the length of b, respectively. From now on we will use only the first letter of a predicate to denote it.

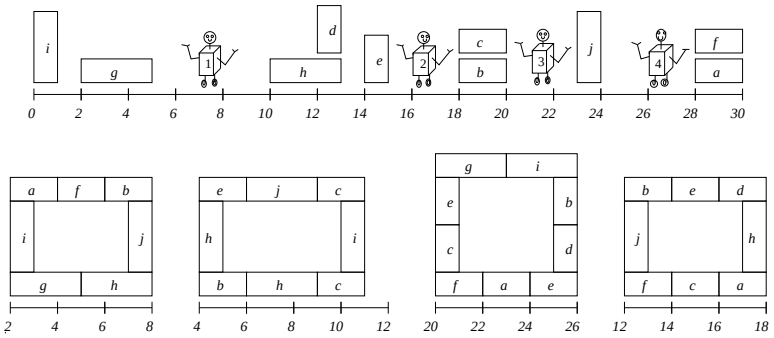

Figure 1: A Sticky Blocks World Example

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:[3]
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:** $O(b,x,z), C(b), C(y), A(a_i,x), A(b,x)$ **dell:** $C(y), A(a_i,x), A(b,x)$, **add:** $O(b,y,z), A(a_i,y), A(b,y)$]

[2] 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.

[3] We assume in this specific example that all agents have identical capabilities. The operators are indexed with respect to the agent that performs them.

Rotate$_i(b)$—rotate b by $\pm\frac{\pi}{2}$. [**cost:** $l^2(b)$, **prec:** $C(b), A(b,y), A(a_i,y), O(b,x,z)$ **dell:** $O(b,x,z)$, **add:** $O(b,x,\bar{z}), A(b,y), A(a_i,y)$] (where $\bar{H} = V$ and vise versa)
Move$_i(a_i,x,y)$—Go from x to y. [**cost:** $|x-y|$, **prec:** $A(a_i,x)$ **dell:** $A(a_i,x)$, **add:** $A(a_i,y)$]

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).

More specifically, a base/top may be described as a sequence of adjusted horizontal breaks $\{b_1,\ldots,b_k\}$ such that $r(b_i) = r(b_{i-1}) + l(b_{i-1})$ for all $i > 1$. Therefore, in this example the subgoal may be specified informally in the following way: $\{O(b_1,x,H), O(b_2,x+l(b_1),H),\ldots,O(b_k,r(b_{k-1})+l(b_{k-1}),H) \mid \sum_k l(b_k) \geq 6\}$. Similarly, a column can be described as
$\{O(b_1,x,V), O(b_2,b_1,V),\ldots,O(b_k,b_{k-1},V) \mid \sum_k l(b_k) \geq 3$.

The interface constraints, $D(G)$, may be described as follows: $\{l(col_1) = l(col_2)$,$l(base) = l(top)$, $r(base) = r(col_1) = r(top)$,$r(col_2) = r(col_1) + l(base) - 1$, $(Blocks(col_1) \cap Blocks(col_2) \cap Blocks(base) \cap Blocks(top)) = \emptyset\}$. The intuitive meaning of $Blocks$ is the set of blocks making up the structural component; we constrain different parts to use different blocks.

Given this *a priori* breakdown into subgoals, our agents are to go through a planning process that will result in satisfying the original goal.

3 General Overview of the Main 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 merge them in an optimal (and inexpensive) way.

As mentioned above, constraining the plan to satisfy fixed subgoals may derive a non-optimal plan. Consider again the scenario presented in Section 2.2. 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 optimality of solution returned by our process is limited by this inherent sub-optimality of restricting ourselves to fixed subgoals.

In this section, we describe a cost-driven merging process that results in a coherent global plan, given the sub-plans. We assume that each agent, i, has been assigned (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.

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 the optimal plan is found. Plans are represented as sets of absolutely necessary sets of constraints that enable them.

Essentially, the search method employs an A^* algorithm. The heuristic function ($f' = g + h'$) that guides the search is dynamically determined 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 that each agent assigns to his own sub-plan.

In general h' is an underestimate, since plans will tend to interfere with one another, but due to overlapping constraints ("favor relations" [18], or "positive" interactions) it might sometimes be an overestimate.

The procedure has the following advantages: **(a)** alternatives are generated by the entire group dynamically (allowing the procedure to be distributed [7]); **(b)** conflicts and "positive" interactions are addressed within a unified framework. **(c)** The complexity of the merging process is significantly reduced; First because the branching factor of the search space is strictly constrained by the individual plans' constraints. Second, the A^* algorithm is using 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 for a successor).

3.1 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 needed for any plan to achieve the grounded instance of the subgoal g_i, starting with the initial state g_i^0. In accordance with the partial order over these constraints,[4] 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 $\mid \hat{e}_i(g_i) \mid$.

- A refinement of $\hat{e}_i(g_i)$ is either a different order of that set that would still achieve the goal, or a further specification (extended partition) of its sub-sets that enable them (the sub-sets). $\mathcal{E}_i(g_i)$ denotes the set of all the refinements of $\hat{e}_i(g_i)$.

  We denote the components of $e(g)$ (any member of $\mathcal{E}(g)$) by $\bigcup_j E_g^j$, such that E^j includes all the constraints that can be satisfied within j steps, and are necessary at some step $\geq j$. For any $j \geq \mid e(g) \mid$, we define E^j to be the description of the goal g.
  Example: Assume in the sticky blocks world scenario from Section 2.2, that g_1 is determined to be $O(g,h,H), A(g,11), A(h,11)$. Then $\hat{e}_1(g_1)$ would be: $\{[C(g),C(d)](=\hat{E}_1^1)\cup[A(a_j,r(d)),C(g),C(d)](=\hat{E}_1^2)\cup[C(g),C(h)](=\hat{E}_1^3)\cup[A(a_i,r(h)),C(g),C(h)]\cup[A(h,11)),O(h,11,H),C(g),C(h)]\cup[A(h,11)),O(h,11,H),C(g),C(h),A(a_i,r(g))]\cup[A(h,11)),O(h,11,H),,A(g,11),O(g,h,H)]\}$ (inducing the plan $\langle M(r(a_i),r(d)),T(d,12,9),M(9,10),T(h,10,11),M(11,2),T(g,2,11)\rangle$).

  One refinement of this plan would replace $\hat{e}_1^2$ with $[A(a_j,r(g)),C(g),C(d)]\cup[A(g,9),C(g),C(d)]\cup[A(a_j,r(d)),C(g),C(d)]$ (inducing the plan $\langle M(r(a_i),r(g)),T(g,2,7),M(r(a_i),r(d)),T(d,12,14),M(14,10),T(h,10,11),M(11,7),T(g,7,11)\rangle$).

- $P(E)$ denotes the set of "grounded" plans (what Chapman calls "complete" plans [1]) that is induced by the set of constraints E.

  $F_{ollow}^1(E)$ is defined to be the set of constraints that can be satisfied by invoking at most one operator, given the initial set of constraints E ($F_{ollow}^1(E) = \{I \mid \exists op \exists P[op(P(E)) \models I]\}$).

[4] Constraints are temporally ordered sets of the domain's predicates associated with the appropriate limitations on their codesignation [1].

Similarly $F^2_{ollow}(E)$ is the set of constraints that can be satisfied by invoking at most two operators *simultaneously* (by two agents) given E, and $F^n_{ollow}(E)$ is the set that can be achieved by n simultaneous actions.

- An *interface point* $I(p, \{p_j\})$ of a plan and a set of plans is a constraint that enables the plan and may become (under some bindings or ordering) a conflict point. 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. It will also be called *soft* if there is a temporal order that resolves the conflict with its contradicting constraints. (In the example, the location of each element of the window [actually the location of the window itself] is a soft interface point.)

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

3.2 The Algorithm

This section describes the algorithm in more detail, along with a running example. At each step of the procedure, agents try to impose more of their private constraints on the group's aggregated set of sets of constraints.

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 of set that has the maximal value according to the heuristic function at step k.

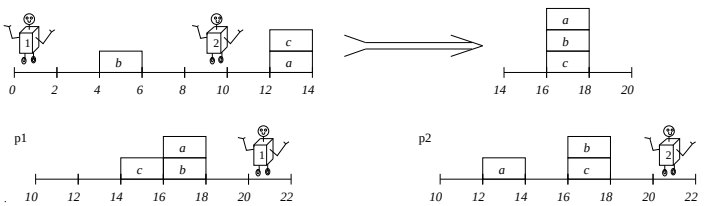

Figure 2: A Simple Sticky Blocks World Example

As an example, consider a simpler scenario of the sticky blocks world as described in Figure 2 (we use the same operators and predicates as described in Section 2.2). 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 $\{A(a_1, 0), A(a_2, 9), A(a, 12), O(a, 12, H), A(c, 12), O(c, a, H), A(b, 4), O(b, 4, H)\}$. The agent's subgoals are (respectively) $g_1 = \{A(a, 16), A(b, 16), O(a, b, H)\}$ and $g_2 = \{A(b, 16), A(c, 16), O(b, c, H)\}$.[5] The lower part of Figure 2 describes the resulting states

[5]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 [15]. Note that since we demand a coherent instance all blocks

that the two corresponding sub-plans would result in, if executed in isolation. To simplify things we will use throughout this example only $\hat{e}_i$ instead of $\mathcal{E}_i$, and $F^1_{ollow}(A^{k+}$ instead of $F^n_{ollow}(A^{k+})$.

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 the initial state g_i^0, and consistent with $D(G)$ and its refinements. The set of alternatives is initialized to be the empty set ($\mathcal{A}^0 = \emptyset$).

   In our example, we have: $\hat{e}_1(g_1) = \{[C(b), C(c)] \cup [A(a_i, r(b)), C(b), C(c)] \cup [C(b), C(c), A(b, 16)] \cup [A(a_i, r(c)), C(b), C(c), A(b, 16)] \cup [C(a), C(b), A(b, 16)] \cup [A(a_i, r(a), C(a), C(b), A(b, 16)] \cup [O(a, b, H), A(a, 16), A(b, 16)]\}$ (this ordered set induces the plan $\langle M(0, 4), T(b, 4, 16), M(16, 12), T(c, 12, x_c = 14 \mid 12), M(x_c, 12), T(a, 12, 16)\rangle$) and $\hat{e}_2(g_2) = \{[C(c), C(b)] \cup [A(a_j, r(c), C(c), C(b)] \cup [C(b), A(c, 16), C(c)] \cup [A(a_j, r(b), C(b), A(c, 16), C(c)] \cup [A(b, 16), A(c, 16), O(b, c, H)]\}$ (inducing the plan $\langle M(9, 12), T(c, 12, 16), M(16, 4), T(b, 4, 16)\rangle$). Note, that $\mathcal{E}_i$ has in general, even in this example, more elements. For example, $\mathcal{E}_1$ would also include the order by which block c is removed from the top of block a before block b is being located at region 16 and $\mathcal{E}_2$ would include the refinement of e_2 in which prior to removing block c, block b is being moved to region 14 (inducing a_2's optimal plan: $\langle M(9, 4), T(b, 4, 14), M(14, 12), T(c, 12, 16), M(16, 14), T(c, 14, 16)\rangle$)

2. At step k one aggregated set A_j^{k+} is chosen from all sets with minimal heuristic value, $\mathcal{A}^{k+}$, and is the one 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+} (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: $(\bigcup_{n=1}^{l} E_i^l \subseteq A_j^{k+}) \wedge (E_i^* \subseteq F^n_{ollow}(A_j^{k+}))$.

   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 $E^1_{g_i}$. In our example, this will be $E^1_{g_1} = E^1_{g_2} = [C(c), C(b)]$.

   At the second step, $A^{1+} = [C(c), C(b)]$, a_1 declares E_1^*, which in this example is equal to $E^2_{g_1} = [A(a_i, r(b)), C(b), C(c)]$. Similarly, a_2 declares $E^2_{g_2} = [A(a_j, r(c), C(c), C(b)]$. (Both are in $F^1_{ollow}(\mathcal{A}^1)$, which contains only one subset.)

   At the third step $A^{2+} = [A(a_i, r(b), A(a_j, r(c), C(c), C(b)]$, a_1 declares $[A(b, 16), C(b), C(c)]$ and a_2 declares $[A(c, 16), C(b), C(c)]$.

   At the fourth step $E_1^* = [A(a_i, r(a), C(a), C(b), A(b, 16)]$ and $E_2^* = [A(b, 16), A(c, 16), O(b, c, H)]$ (a_2's final subgoal).

   At the fifth step both agents declare their final goals which are both satisfied.

are explicitly bounded to be located at region 16. Even if this were not the case, however, the problem would still be solvable.

3. At this step, $\{Ex_r(A_j^{k+})\}$, all the maximal consistent 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: $\{I \mid (I \in \bigcup_i E_i^*) \wedge (I \cup Ex(A_j^{k+}) \not\models F_{alse})\}$).

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

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

   At the third step, A^{2+} has 4 possible extensions: $Ex_1 = [A(c, 16), C(b), C(c)]$, $Ex_2 = [A(b, 16), C(b), C(c)]$, $Ex_3 = [A(b, 16), C(b), A(c, 16)]$ (where b is located on a), and $Ex_4 = [A(b, 16), C(b), A(c, 16)]$ (where b is located on a). Thus, $\mathcal{A}^3$ has four different members (the union of A^2 with each of the extensions).

   At the fourth step, $A^{3+}(= A_3^3)$ has one extension: $[A(a_i, r(a), C(a), C(b), A(b, 16), A(c, 16), O(b, c, H)]$ which includes both agents' declarations.

   At the fifth step, the sole extension of A^{4+} is the description of both subgoals.

4. At this stage, based on the extended set of constraints, the agents construct additions to the ongoing candidate plans. Each extension that was generated in the previous step induces a sequence of operations that achieves it ($P_r(Ex_r(A_j^{k+}))$). The generation of these sequences is fully described in Section 3.3.

   The corresponding alternative multi-agent plans that have been generated so far are concatenated with The resulting segments of plans: $P(A_j^{k+}) \parallel \{P(Ex_r(A_j^{k+}))\}$. Thus, the g value of alternative extended sets of constraints is determined.

   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 be achieved by $\langle\{M_1(0, 4), M_2(9, 12)\}\rangle$, yielding a g value of $7(= 4 + 3)$. The four possible extensions that are generated in the third step can be achieved (respectively) by the following plans $\langle T_1(b, 4, 16)\rangle$, $\langle T_2(c, 12, 16)\rangle$, $\langle\{T_1(b, 4, 16), T_2(c, 12, 16)\}\rangle$, and $\langle\{T_2(c, 12, 16), T_1(b, 4, 16)\}\rangle$. Thus, $g(A_1^4) = 29(= 7 + 22)$, $g(A_2^4) = 29$,$g(A_3^4) = g(A_4^4) = 37$. At the fourth step the extension is achieved by $\langle M_i(12, 16)\rangle$ yielding a g value of 41. The final extension is established by $\langle T_i(a, 12, 16)\rangle$ and the corresponding g value equals the cost of the entire plan which is 45.

5. All extensions are evaluated so as to direct the search (i.e., find the h' component of the A^* search). Each agent declares, h_i', the estimate it associates with each newly-formed set of aggregated constraints (the cost of completing "his private" sub-plan). 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 to determine the h' value by each agent. 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 "interactive 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.)

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 99). This value remains the same at the second step since both agents carry out the first step of their initial plans. At the third step, the four possible extensions score respectively the following f' values according to the "rigid policy": $89, 89, 77, 82$. According to the "interactive policy", the scores are $71, 83, 49, 73$.

As an example consider A_1^4: This set fully satisfies $E_1^1 \cup E_1^2 \cup E_1^3$. Therefore, according to both policies, a_1 gives it the heuristic value of 18. But according to a_2's evaluation, only $E_2^1 \cup E_2^2$ are satisfied. Therefore, a_2 declares the estimated cost of achieving his final subgoal, given only this subset of his full set of constraints. According to his original plan, this cost is 42; actually re-planning to achieve its final goal, given A_1^4, would yield the plan $\langle T(c, 12, 14), M(14, 16), T(b, 16, 18), M(18, 14), T(c, 14, 16), M(16, 18), T(b, 18, 16)\rangle$ which actually costs only 24.

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 when all "best plans" have been found. Since the heuristic function is not guaranteed to be an underestimate, stopping when the first global plan has been generated may not result in the optimal global plan (i.e., another path, with higher h' 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.

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

3.3 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 moves are determined. We require that the actual plan will be constructed dynamically in order to determine the g value of each alternative. Note, however, that it would be possible to use Best First Search instead to determine the entire set of constraints, and only then construct the induced plan. Employing such a policy 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 achieves 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 consumed space would become much larger.

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 for 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_{ollow} of the already achieved set. Therefore, it is straightforward to detect actions that can be taken in parallel. Thus the plan that is constructed is not just cost efficient, but also time efficient. In this framework, 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 the utility is a function of execution time *and* cost.

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

In the example all the suggested methods would yield 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 $Ex(A^1) = [A(a_i, r(b), A(a_j, r(c), C(c), C(b)]$. These constraints can be achieved by $M_i(r(a_i), r(b))$ and $M_j(r(a_j), r(c))$. The bids that a_1 and a_2 give to these actions are respectively $[4, 12]$ and $[6, 3]$. Therefore, a_1 is "assigned" to block b and a_2 is assigned to block c. The constructed plan is $\langle\{M_1(0, 4), M_2(9, 12)\}\rangle$ yielding a g value of $7(= 4 + 3)$ where both agents perform in parallel.

Based on the agents' bids, the four possible extensions that are generated in the third step can be achieved (respectively) by the following plans $\langle T_1(b, 4, 16)\rangle$, $\langle T_2(c, 12, 16)\rangle$, $\langle\{T_1(b, 4, 16), T_2(c, 12, 16)\}\rangle$, and $\langle\{T_2(c, 12, 16), T_1(b, 4, 16)\}\rangle$. Thus, there are four possible "ongoing" plans. Note that there is a scheduling difference between the third and fourth extensions. According to the first, a_2 places block c at region 16 and only then a_1 places b on c; following the second plan, a_2 should wait for a_1 to accomplish its mission. At the fourth step, the extension is achieved by $\langle M_i(12, 16)\rangle$. Either agent may be assigned this task. The final extension is established by $\langle T_i(a, 12, 16)\rangle$, and the corresponding final plan that is constructed is: $\langle\{M_1(0, 4), M_2(9, 12)\}, \{T_2(c, 12, 16), T_1(b, 4, 16)\}, M_i(12, 16), T_i(a, 12, 16)\rangle$

Theorem 1 *Given the set $\{p_i\}$ of 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 $O(q \times l)$ steps where l is the length of the optimal plan, and q is the number of positive interactions between overlapping constraints. 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 length of a completed plan.*

Proof (Sketch): Let P^* be the optimal multiagent 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 an underestimate.

The proof is by induction on the subsets of constraints, $e^1(G) \cup e^2(G) \cup \ldots \cup e^n(G)$, 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 $e_i^1(g_i) \in \mathcal{E}_i(g_i)$ for some a_i, and each of the corresponding agents, a_i, will be able to declare in the first step all elements of $\mathcal{E}_i(g_i)$. Thus, $e^1(G)$ is guaranteed to be generated.

2. Assume that the claim holds for $e^j(G) \mid j = 1, \ldots, n$. We have to prove that $e^n(G)$ will be found. According to the induction assumption, $e^{n-1}(G)$ will be reached at some step r. According to the underestimate assumption it will be chosen at step $r+1$ as the best alternative for expansion. Since $e^n(G) \models G$ and belongs to $F^n_{ollow}(e^{n-1}(G))$, it must be the case that each g_i belongs to $F^n_{ollow}(e^{n-1}(G))$. Thus, each a_i would declare g_i at step r, and $e(G)$ will be found.

The number of steps in our search is bounded by $O(q \times l)$. At any step of the procedure, the best candidate step is being maximally extended. Since we may assume the heuristic function is an underestimate (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. □

4 Finding Coherent Instances of Subgoals

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 the example given in Section 2.2). 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 ($\{g_i\}$), and **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 domains axioms). In this step, we care only about unbridgeable conflicts, such as the final resources that are being used for the final subgoals. Soft interface points are being 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 being 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 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 sub-goals. Since such global knowledge is not alway available we suggest having the 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 being 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. Or (at each iteration of the search) have the plans generated in the first step serve as constraints (fully or partially) for 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 2.2, two possible coherent instances are found after two iterations of the first step, as shown in Figure 3. The first coherent instance is chosen. The rest of the example is worked out in [8].

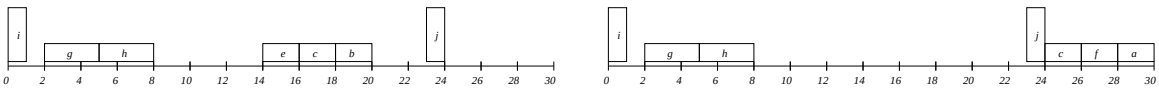

Figure 3: Coherent Combination of Instances

5 The Anytime Planning Approach

As described in the previous section, the merging process is 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 framework is not guaranteed. Obviously, the amount of time and reasoning invested in 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 the how close to an optimal plan the system designer wishes 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, in 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 [3, 19]. Figure 4 describes the performance profile of the process (the step function lines). The dotted line describes a "diminishing returns" type decision procedure, to which the 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 P^* line denotes the optimal plan generated for a given subgoal setup, which is in general inferior to the absolute optimal plan described by $s(P)^*$.

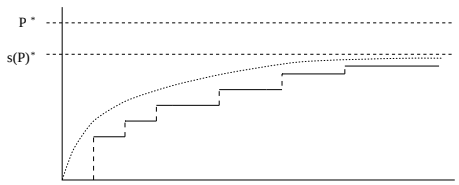

Figure 4: Performance Profile of the Process

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.

6 Non-Benevolent Agents—Vickrey's Mechanism

Throughout this paper we have assumed that the agents are benevolent. If agents are self-interested (utility maximizers) they should be motivated to contribute to the global plan. In such a case, agents should be paid for their work. Thus, each agent should be paid for performing an operator according to the bid it gives for that operator.

In constructing an optimal plan, it is critical that agents, at each step, express their true cost values (bids). However, if our group consists of autonomous, self-motivated agents, each concerned with its own utility (and not the group's welfare), they might be tempted to express false cost values, in an attempt to gain higher payment. This is a classic problem in voting theory: the expression of cost values at each step can be seen as an (iterative) cardinal bidding procedure, and we are interested in a non-manipulable bidding scheme so that the agents will be kept honest.

Fortunately, there do exist solutions to this problem, such that the above plan choice mechanism can be used even when the agents are not necessarily benevolent and honest. By using a variant

of the Vickrey Mechanism [16], more commonly known as the "Dutch Auction," it is possible to ensure that all agents will bid honestly. This is done by minor changes to Step 4 of the procedure given in Section 3.2. As before, an operator would be assigned to the the agent that declares the minimal cost (bid), however the actual payment would be equal to the next lowest bid. Note, that under this scheme total payment for the plan would be greater than its actual cost.

To see why declaring the true cost is the dominant strategy consider a_i facing a bid over Op_k. Say 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 $\geq X$ than 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). A formal analysis of the process supports the following claim:

Theorem 2 *At* any *step k of the procedure, i's best strategy is to bid over the induced actions at that step (all operators in $P(Ex(A^{k+}))$ according to his true cost.*

7 Related Work

An approach similar to our own is taken in [13] 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 than merging these plans. The basic idea there is to try and make a global plan by repeatedly merging plans that achieves the separate subgoals. Finding the solution is guaranteed (under several restrictions) only if a sufficient number of alternative plans are generated for each subgoal. Our approach does away with the need for several plans by treating constraints instead of sequences of actions. We also have agents do the "merging" in parallel.

Our approach also resembles the GEMPLAN planning system [12]. 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).

8 Conclusions

In this paper, we presented a novel multi-agent planning procedure. 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.

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.

References

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

[2] 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.

[3] 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.

[4] E. H. Durfee. *Coordination of Distributed Problem Solvers.* Kluwer Academic Publishers, Boston, 1988.

[5] E. H. Durfee and V. R. Lesser. Using partial global plans to coordinate distributed problem solvers. *Proceedings of the 1987 IJCAI*, pages 875–883, 1987.

[6] E. H. Durfee and V. R. Lesser. Negotiating task decomposition and allocating using partial global plannning. In A. H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence, Vol. II*, pages 229–243. Morgan Kaufmann, San Mateo, California, 1989.

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

[8] E. Ephrati and J. S. Rosenschein. Multi-agent planning through exploitation of subgoals. Technical report, Computer Science Department, Hebrew University, Jerusalem, Israel, 1993. In preparation.

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

[10] Matthew Katz and Jeffrey S. Rosenschein. Verifying plans for multiple agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 1993. To appear.

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

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

[13] 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.

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

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

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

[17] F. von Martial. Multiagent plan relationship. In *Proceedings of the Ninth International Workshop on Distributed Artificial Intelligence*, pages 59–72, Rosario Resort, Eastsound, Washington, 1989.

[18] F. von Martial. Coordination of plans in multiagent worlds by taking advantage of the favor relation. In *Proceedings of the Tenth International Workshop on Distributed Artificial Intelligence*, Bandera, Texas, October 1990.

[19] S. Zilberstein and S. J. Russell. Effecient 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.