Solving Complex Planning Tasks Through Extraction of Subproblems

Jana Koehler
Institute for Computer Science
Albert Ludwigs University
Am Flughafen 17
79110 Freiburg, Germany
koehler@informatik.uni-freiburg.de

Abstract

The paper introduces an approach to derive a total ordering between increasing sets of subgoals by defining a relation over atomic goals. The ordering is represented in a so-called *goal agenda* that is used by the planner to incrementally plan for the increasing sets of subgoals. This can lead to an exponential complexity reduction because the solution to a complex planning problem is found by solving easier subproblems. Since only a polynomial overhead is caused by the goal agenda computation, a potential exists to dramatically speed up planning algorithms as we demonstrate in the empirical evaluation.

Introduction

How to effectively plan for interdependent subgoals has been in the focus of AI planning research for a very long time (Chapman 1987). But until today planners have made only some progress to solve larger sets of subgoals and scalability of classical planning systems is still a problem.

Previous approaches fell into two categories: On one hand, one can focus on the detection of conflicts caused by goal interdependencies to guide a planner during search, e.g., (Drummond & Currie 1989; Hertzberg & Horz 1989). On the other hand, one can perform a structural analysis of the planning task to determine an appropriate goal ordering before planning starts, see (Irani & Cheng 1987; Cheng & Irani 1989; Joslin & Roach 1990).

Unfortunately, finding the "right" subgoal ordering is as hard as solving the planning problem at hand and thus either the computation is infeasible for larger problem instances (Joslin & Roach 1990) or severe restrictions need to be put on the representation (Cheng & Irani 1989).

The approach we are going to describe falls into the second category. It implements a preprocessing phase that allows the planner to gain insight into structural properties of a planning problem before planning starts. In terms of (Korf 1987; Barrett & Weld 1993) it introduces a method to efficiently determine an order for laboriously serializable subgoals in many cases.

The general idea can be summarized as follows: Given a set of atomic goals, the aim is to compute a *goal agenda* as a total ordering between increasing subsets of goal atoms. Such an agenda can be used in various ways during planning. In this paper, we just propose one way, namely that of incrementally planning for the increasing goal subsets from the agenda. This way, the efficiency of planning can be dramatically improved in many cases while preserving the completeness of the planning algorithm.

To compute the goal agenda, a so-called *goal analysis* is activated, which is based on a relation $\prec$ between atomic goals. In the paper, we are going to present one possible definition for $\prec$, but in general one can imagine many other domain-specific or domain-independent relations. We only put a restriction on them in terms of computational complexity: the aim is to invest some polynomially bounded overhead in the hope to gain an exponential complexity reduction.

The $\prec$ relation defines a directed graph with the goal atoms as vertices and one edge per valid relation. From this graph, an ordering over increasing subsets of goal atoms is derived. If the ordering is total, the goal agenda is immediately obtained. If only a partial ordering over the sets is obtained, the goal analysis tries to order the remaining unordered sets. If it succeeds, a goal agenda is returned, otherwise, the goal agenda collapses into one single set. This means, the $\prec$ relation as defined fails in characterizing the particular planning domain (or a specific class of planning problems in a given domain) and should be replaced by a different definition for $\prec$. Note that in this approach, we can derive a total ordering over subsets of goals without considering the exponentially many possible subsets of a given goal set.

Defining a Relation between Atomic Goals

As usual, planning tasks are specified by a set of operators, a finite set of domain constants, the initial state and the goal. The initial state and goals are conjunctions of ground atoms. For the representation of operators we consider a language that allows conditional effects. An action o is a ground instance of an operator and has the syntactic form

$$\begin{aligned} o: \quad & \mathrm{pre}_0(o) \\ & \phi_0 : \mathrm{eff}_0^+(o), \mathrm{eff}_0^-(o) \\ & \phi_1 : \mathrm{pre}_1(o) \Rightarrow \mathrm{eff}_1^+(o), \mathrm{eff}_1^-(o) \\ & \vdots \\ & \phi_n : \mathrm{pre}_n(o) \Rightarrow \mathrm{eff}_n^+(o), \mathrm{eff}_n^-(o) \end{aligned}$$

The precondition of the action is denoted with $\mathrm{pre}_0(o)$ and its (unconditional) positive and negative effects with $\mathrm{eff}_0^+(o)$ and $\mathrm{eff}_0^-(o)$. Each conditional effect ϕ_i consists of an effect condition $\mathrm{pre}_i(o)$, and the positive and negative effects $\mathrm{eff}_i^+(o)$ and $\mathrm{eff}_i^-(o)$.

Given a set of goals, various ways to define a relation over them exist. In principle, one can distinguish between *domain-dependent* and *domain-independent* definitions for $\prec$. For example, in a scheduling domain, a domain-dependent definition for $\prec$ could read: Given two goals A and B, if $A = paint(x)$ and $B = polish(y)$ and $x = y$ then $B \prec A$.

Although domain-dependent definitions can be very effective, they need to be redeveloped for each single domain. Therefore, we are in particular interested in domain-independent definitions having a broader range of applicability.

One possible domain-independent definition for $\prec$ can be derived from the following scenario: Given two atomic goals A and B let us assume the planner has reached a state s_A satisfying A, but in which B does not hold. Does there exist a plan π constructed from a given set of actions $\mathcal{O}$, which

1. can be executed in s_A
2. reaches a state in which B holds
3. but leaves A valid in any state that will be traversed during the execution of π?

A positive answer does not seem to imply any problems for the planner, but a negative answer would imply that any plan starting in s_A and achieving B had to destroy and reestablish A. Therefore, it seems to be reasonable to first plan for B and then to plan for A if the question is answered with "NO".

Definition 1 *Given two goal atoms A and B, and a set of actions $\mathcal{O}$ the relation $B \prec_1 A$ holds if and only if no plan exists over $\mathcal{O}$ that*

1. is executable in any state in which A holds

2. achieves B

3. leaves A always valid during its execution.

This definition gives $B \prec_1 A$ the intuitive meaning that if there is evidence that B cannot be achieved after A has been achieved then B is a goal prior to A. Although, this seems to be a promising candidate for the definition of $\prec$, it seems to be much too difficult to test, since it requires to show that *all* possible plans over the given action set $\mathcal{O}$ will violate at least one of the conditions. We now proceed in three steps

1. We can exploit more knowledge about the states s_A in which A has been made true.
2. We can restrict the set of actions $\mathcal{O}$ that needs to be considered to form plans π.
3. We develop an easy to verify test.

Exploiting Knowledge about the State s_A

When trying to compute a goal agenda, the planner is faced with a specific planning problem involving an initial state I, a goal set $G = \{A, B, \ldots\}$, and the set $\mathcal{O}$ of all possible ground instances of all operators. The state s_A denotes a kind of "generic state" and represents all possible states that are reachable from I and in which A has been made true, i.e., $s_A \models A$. More interesting for the planner, however, is to find out which atoms can never hold together with A, i.e., do determine a set F with $s_A \not\models \mathsf{F}$. This set can be easily computed using planning graphs (Blum & Furst 1997). Starting from I with $\mathcal{O}$, the planning graph is built until it reaches its *fixpoint*. All atoms, which are marked as exclusive of A in the fixpoint level of facts belong to F because no state in the world can make these exclusive facts true independently of how many actions will be executed.

$$\mathsf{F} = \{p \mid p \text{ is exclusive of } A \text{ in the fixpoint level}\}$$

Note that the planning graph is only grown once for a given I and $\mathcal{O}$, but can be used to determine the F sets for all atomic goals in G (and arbitrary subsets of G). But since even indirectly building planning graphs takes a rather long time, we have developed an alternative method that analyses the planning actions directly.

Direct operator analysis works as follows: F contains all facts that are always made false when A is made true. This means for a single action that achieves A unconditionally that its unconditional negative effect $\mathrm{eff}_0^-(o)$ would belong to F. If it achieves A with a conditional effect ϕ_i, then its unconditional negative effect $\mathrm{eff}_0^-(o)$, the conditional negative effect $\mathrm{eff}_i^-(o)$, and all conditional negative effects $\mathrm{eff}_j^-(o)$ whose effect condition is a subset of the effect condition of ϕ_i would belong to F. Consequently, for a set of actions the intersection over these negative effects needs to be determined, which contains all atoms that will be made false if A is made true independently of the particular action effect that is used.

$$\begin{aligned} \mathsf{F} &= \bigcap_{i, o \in \mathcal{O}} M_i(o) \\ M_i(o) &= \bigcup_{\substack{pre_j(o) \subseteq pre_i(o) \\ A \in \mathrm{eff}_i^+(o)}} \mathrm{eff}_j^-(o) \end{aligned}$$

Both methods have its advantages. In general it holds that using planning graphs is more costly in terms of runtime and memory, but leads to larger F sets for domains represented with STRIPS operators. Direct operator analysis is much faster, needs less memory, and leads to smaller F sets in the case of STRIPS, but significantly larger F sets when ADL operators are used. In all tests, both methods always resulted in the same goal agendas. Some illustrating examples can be found in (Koehler & Hoffmann 1998). In the empirical tests, direct operator analysis was used to determine F.

The set F can now be used to exclude some actions from the set $\mathcal{O}$.

Restricting the action set $\mathcal{O}$

The condition that no plan is allowed to make A false during its execution limits the set of actions from which valid plans can be constructed: no effect ϕ_i with $A \in \text{eff}_i^-(o)$ can be used in the plan. This means, we can eliminate all actions that unconditionally delete A or that will not be applicable in s_A since they require atoms from the F set as preconditions. Furthermore, all "bad" conditional effects that delete A or whose effect conditions include atoms from the F set are discarded from the remaining actions. This yields a restricted action set $\mathcal{O}^R$ in which all inapplicable or violating effects have been removed from each action description o.

$$\mathcal{O}^R = \{red(o, \mathsf{F}, A) \mid A \notin \text{eff}_0^-(o) \text{ and } \text{pre}_0(o) \cap \mathsf{F} = \emptyset\}$$

with $red : o \times \mathsf{F} \times A \mapsto o'$ such that

$$\begin{array}{rcl}
\text{pre}_0(o') & = & \text{pre}_0(o) \\
\text{eff}_0^+(o') & = & \text{eff}_0^+(o) \\
\text{eff}_0^-(o') & = & \text{eff}_0^-(o) \\
\phi_i(o') & = & \left\{ \begin{array}{ll} \phi_i(o) & \text{if } \text{pre}_i(o) \cap \mathsf{F} = \emptyset, A \notin \text{eff}_i^-(o) \\ \emptyset & \text{otherwise} \end{array} \right.
\end{array}$$

For example, given $\mathsf{F} = \{b, c\}$, $A = a$, and an action with precondition x, unconditional effects w, $\neg\, z$, and conditional effects ϕ_1: $g,h \Rightarrow \neg\, a$ and ϕ_2: $b \Rightarrow y$, just the unconditional remainder $x \Rightarrow w, \neg\, z$ would be added to $\mathcal{O}^R$ because ϕ_1 deletes the goal a and ϕ_2 is inapplicable because of $b \in \mathsf{F}$.

Depending on how the F set was obtained, two different computations are performed. If planning graphs have been used, $\mathcal{O}^R$ is the reduced action set that will be used for the goal agenda computation, because it contains all actions that are applicable in any state s reachable from s_A with $s \models A$ and that leave A valid.

If the direct operator analysis has been used, a fixpoint computation on $\mathcal{O}^R$ is necessary, because actions from $\mathcal{O}^R$ could possibly make atoms from F true, i.e., less actions can be excluded. In the case of planning graphs, this fixpoint computation is not necessary because even if actions from $\mathcal{O}^R$ could possibly make atoms from F true, these atoms can never hold in a state satisfying A and we have required that A remains valid after it has been achieved in s_A.

For the fixpoint computation we test if there exists an atom $p \in \mathsf{F}$ with $p \in \text{eff}_i^+(o)$ where $o \in \mathcal{O}^R$. If this is the case, we determine $\mathsf{F} = \mathsf{F} \setminus \{p\}$ and repeat the reduction operation over $\mathcal{O}$ to obtain a larger set $\mathcal{O}^R$. This process is repeated until F remains unchanged and the final set $\mathcal{O}^R$ results from it.

Lemma 1 *Given the action set $\mathcal{O}^R$, each possible plan π that can be formed using actions from this set satisfies that if the goal A was true in the state in which π is executed then it will remain true during the execution of π.*

The proof is obvious, since all harmful action effects have been eliminated. The lemma implies that the third condition in Definition 1 is automatically satisfied when we consider $\mathcal{O}^R$ instead of $\mathcal{O}$, which yields a simplified definition $\prec_2$:

Definition 2 *Given two goal atoms A and B, and a set of actions $\mathcal{O}^R$, $B \prec_2 A$ if and only if for all plans π over $\mathcal{O}^R$ holds that*

1. *B does not hold in s_A and*
2. *if π does achieve B then it is not executable in s_A.*

Since the remaining two conditions are still too difficult to test for *all* possible plans, we have to approximate them with easy-to-test criteria.

Deriving an Approximative Test

Based on Definition 2 the goal analysis had to perform two tests:

1. B does not hold in s_A and
2. If a plan achieves B then it is not executable in s_A.

Apparently, the first condition cannot be effectively tested as it would require to investigate all, i.e., infinitely many plans that can lead to s_A. Surprisingly, it turned out that simply assuming $s_A \not\models B$ is a very powerful heuristic, i.e., the condition is not tested at all and only the second test is performed.

To verify the second condition, it is tested that no action in $\mathcal{O}^R$ has B as positive effect. In this case, no plan can achieve B. Otherwise, it is shown that each action that has B as a positive effect, has at least one precondition that cannot be achieved by a plan that can be formed over $\mathcal{O}^R$. Note that this is a recursive instance of the original problem. We show that B cannot be achieved after A by showing that whenever an action could make B true then one of its preconditions cannot be achieved after A. To avoid infinite regression over preconditions of actions, we only test that there exists at least one precondition p that is not a positive effect of any action in $\mathcal{O}^R$ and that does not hold in s_A. Again, $s_A \not\models p$ is simply assumed to be true. This leads to the following definition for the $\prec$ relation over atomic goals:

Definition 3 *Given two goal atoms A and B, and a reduced set of actions $\mathcal{O}^R$, $B \prec_3 A$ if and only if*

$$\begin{array}{c}
\forall\, o \in \mathcal{O}^R \;\; \forall\, \phi_i(o) : \; \textit{if } B \in \textit{eff}_i^+(o) \textit{ then} \\
\exists\, p \in \textit{pre}_0(o) \cup \textit{pre}_i(o) \, \forall\, o' \in \mathcal{O}^R \;\; \forall\, \phi_j(o') : \\
p \notin \textit{eff}_j^+(o')
\end{array}$$

This definition leads to a sufficient test for the second condition. If the assumptions $s_A \not\models B$ and $s_A \not\models p$ are correct, then $\prec_3$ implies $\prec_2$, which again implies $\prec_1$.

As a simple example, consider the two goals *on(a,b)* and *on(b,c)* together with the following two blocksworld actions that can achieve the goals:

stack(a,b): *clear(b), holding(a)*
$\Rightarrow$ *arm-empty, clear(a), on(a,b)*
$\neg$ *clear(b),* $\neg$ *holding(a).*

stack(b,c): *clear(c), holding(b)*
$\Rightarrow$ *arm-empty, clear(b), on(b,c)*
$\neg$ *clear(c),* $\neg$ *holding(b).*

Assuming that *on(a,b)* has been achieved, is it still possible to stack block b on c with a on top of b? Using planning graphs yields $\mathsf{F} = \{holding(a), clear(b), holding(b), on\_table(a), \ldots\}$ because all these facts are exclusive of *on(a,b)*. Thus, all actions requiring one of the atoms as precondition are excluded from the action set as well as all actions that would make *on(a,b)* false. The only action **stack(b,c)** that can possibly achieve *on(b,c)* is excluded since its precondition *holding(b)* can never be achieved without destroying *on(a,b)* because the two facts are exclusive. Consequently, the relation *on(b,c)* $\prec$ *on(a,b)* is derived. Using direct operator analysis results in $\mathsf{F} = \{holding(a), clear(b)\}$, i.e., a much smaller set, which excludes less actions and therefore **stack(b,c)** is still contained in $\mathcal{O}^R$. But when performing the test from Definition 3, the planner realizes that its precondition *holding(b)* cannot be achieved because all potential actions require *clear(b)* as precondition and have therefore been excluded from $\mathcal{O}^R$.

Computing the Goal Agenda

The result of the goal analysis is a set of relations between pairs of atomic goals which defines a directed graph where vertices are goal atoms and an edge from a vertex A to a vertex B exists if and only if $A \prec B$. For each pair of goals, the analysis returns one of the following possible outcomes:

1. $A \prec B$, but not $B \prec A$
2. $A \prec B$ and $B \prec A$
3. neither $A \prec B$ nor $B \prec A$

In the first case, it seems to be reasonable to derive the ordering $A < B$, i.e., to plan first for A and then for B. In the second, the relation leads to a cycle between the pair and it seems to be reasonable to form one set containing A and B, i.e., to plan for the goal $A \wedge B$. In the third case, no information was derived by the goal analysis and both atoms have to remain unordered. In terms of the goal agenda, the first case leads to the agenda $[1 : A, 2 : B]$, while the second yields an agenda with just one entry $[1 : A \wedge B]$. In the third case, no agenda is returned.

Let us consider the goal set A, B, C, D, E and the relations $A \prec B$, $B \prec C$, and $B \prec D$. First, the relations between atoms can be immediately extended to increasing sets, i.e., if B has to be achieved after A then also any conjunctive goal containing B such as $A \wedge B$ or $A \wedge B \wedge C$ has to be achieved after A. Thus $A \prec B$ implies $A \prec A, B \ldots$. This yields the graph nodes shown in Figure 1.

Second, the $\prec$ relation is obviously transitive when considering such increasing subsets of goals, i.e., if $A \prec A, B$ and $A, B \prec A, B, C$ then $A \prec A, B, C$. The basis for the goal agenda is now the transitive closure of this directed graph, which is found in time less than cubic in the number of nodes in the graph.

For each node n in the transitive closure of the graph the number of ingoing and outgoing edges n^{in}_{out} is counted. All disconnected nodes n^0_0 are moved into a separate set of goals *G-sep* containing now the atomic goals for which no $\prec$ relation was derived. For the remaining nodes n, their *degree* (an integer value) as the difference between the number of ingoing edges and the number of outgoing edges is determined. Nodes with the same degree are assigned to the same set, i.e., one set is obtained per integer value. Now the sets are ordered based on the assigned integer value. This yields a total ordering among increasing subsets of goals (except the *G-sep* set) that has the following properties:

- Sets on each path in the graph are monotonically increasing.
- Nodes that occur in a cycle based on $\prec$ belong to the same set.
- If the original graph contained a path from A to B, but no path from B to A, then the set containing A is ordered before the set containing B.

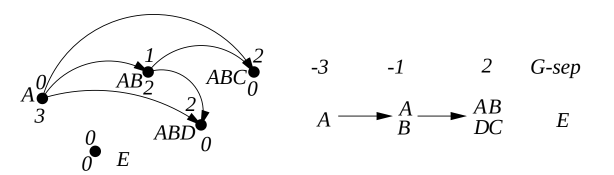

Figure 1: On the left, the in- and outdegrees of the nodes are shown. The node E becomes a member of the *G-sep* set, while the remaining nodes are assigned to three increasing and totally ordered sets based on their degree. With an empty *G-sep* the goal agenda were $[1 : A, 2 : A \wedge B, 3 : A \wedge B \wedge C \wedge D]$.

The problem is the set *G-sep*. If it is empty, then the goal agenda is obtained from the ordered sets. If *G-sep* is non-empty, we have an agenda for a subset of goals but have failed in deriving a complete goal agenda just from analyzing pairs of goal atoms. However, it might still be possible that a complete agenda is successfully computed by comparing subsets of goals with each other. In other approaches, the problem was always which of the exponentially many subsets should be considered in order to avoid exponential overhead. The computed partial agenda offers one possible answer. It suggests taking the set *G-sep* and trying to order it with respect to the sets emerging from the graph. In order to do so, we need to extend the $\prec$ relation to sets of goal atoms instead of single atoms.

Extending the $\prec$ Relation to Goal Sets

First, we need to modify the definition of F and $\mathcal{O}^R$ with respect to a set of goals $\{A_1, \ldots, A_n\}$ and a set of goals $\{B_1, \ldots, B_k\}$ that need to be ordered. The set F' is the set of all atoms that are exclusive to the fact set $\{A_1, \ldots, A_n\}$ in the planning graph. The extension of exclusivity from pairs of atoms to arbitrary sets of atoms is straightforward.

$$\mathsf{F}' = \{p \mid p \text{ is exclusive of } \{A_1, \ldots, A_n\}\}$$

Similarly, direct operator analysis is extended to sets by taking the union over the F sets that were determined for the individual A_i.

F' is usually a much larger set than F and we can expect that it will exclude more actions, i.e., it is more likely that sets can be ordered even if the goal analysis failed for their single elements. Based on F' the action set can be reduced by excluding all actions that make *any* of the goals A_i false or require a precondition from F' to be true. Similarly, the reduction function removes all conditional effects from the remaining actions that can make any of the A_i false or require one of the F' atoms as an effect condition and we end up with a set of actions $\mathcal{O}^{R'}$ reduced wrt. F'. To order goal sets Definition 2 is extended to sets:

Definition 4 *Given two goal sets $\{A_1, \ldots, A_n\}$ and $\{B_1, \ldots, B_k\}$, and a set of actions $\mathcal{O}^{R'}$. The relation $\{B_1, \ldots, B_k\} \prec_S \{A_1, \ldots, A_n\}$ holds if and only if for all plans π over $\mathcal{O}^{R'}$ holds that there exists a $B_i \in \{B_1, \ldots, B_k\}$ such that $s_{\{A_1,\ldots,A_n\}} \not\models B_i$ and if π could achieve B_i then it is not executable in $s_{\{A_1,\ldots,A_n\}}$.*

The corresponding test is based on Definition 3 with the individual B_i as input. The sets are ordered if the test succeeds for at least one B_i as in the case of the example, see Figure 2.

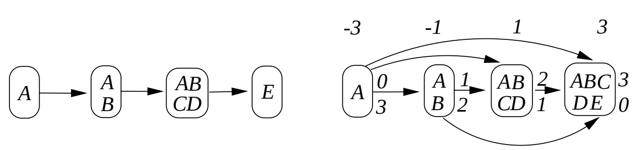

Figure 2: On the left, a directed graph based on the sets and *G-sep* showing the existence of a relation $\{A, B, C, D\} \prec_S \{E\}$. Note that a goal analysis between singleton sets does not need to be repeated. On the right, the transitive closure showing in- and outdegrees and the successfully computed goal agenda for the example based on the totally ordered sets.

Definition 4 defines a graph with the various sets as vertexes and the $\prec_S$ relation as edges. For this graph, the same computation as above is repeated. If it yields a total ordering over all vertexes, the goal agenda contains an entry for each vertex (ordered by their degree) and each entry contains a subset of goal atoms. If the computation yields only a partial ordering, the goal agenda *collapses* into just one entry containing the original goal description, since ordering the remaining unordered sets failed. There is no indication, which other subsets one should try to order and trying arbitrary subsets does not seem to make much sense. A collapse of the goal agenda can be interpreted as a failure of the definition for $\prec$ to characterize constraints between goals, because either no such constraints exist as seems to be the case for example in all variations of logistics problems or because the definition of $\prec$ is inadequate and one should use a different domain-independent or domain-dependent relation.

Goal Agenda Planning

Given a goal agenda with more than one entry, the question is how to use it during planning. Several possibilities exist: First, one can use this information to guide any generative planning algorithm in its choice of the next goal that it has to achieve. Here, the goal agenda provides control information similar to strategies such as ZLIFO (Gerevini & Schubert 1996) or LCFR (Pollack, Joslin, & Paolucci 1997). Second, one can imagine to reuse plans for subproblems stated by the agenda and plan from second principles (Koehler 1996). Finally, one can plan from scratch for each of the subproblems and compose the solution plan from the individual subplans. We are going to explore the latter possibility.

For a goal agenda $[1 : goal_1, 2 : goal_2, \ldots, n : goal_n]$, where each of the $goal_i$ can represent a set of atomic goals satisfying $goal_{i-1} \subseteq goal_i$, the main task is to determine the initial state for a given $goal_i$ as the result of the plan solving $goal_{i-1}$. This problem is trivial for STRIPS, but becomes difficult for more expressive languages. When generating parallel plans for a language that allows conditional effects in actions such as in IPP (Koehler *et al.* 1997) the result of such plans is no longer a single unique state. In fact, each linearization of a parallel plan can lead to a different resulting state and only their intersection, i.e., only the atoms that are true in all resulting states form the new initial state.

The computation of the resulting state follows exactly the semantics of parallel ADL plans as defined for IPP (Koehler *et al.* 1997). Given a unique initial state and a set of actions with conditional effects, one possible linearization is generated. For the first action in the linearization, the effect conditions are evaluated in the initial state. Then all positive effects with satisfied conditions are added to and all negative effects with satisfied conditions are deleted from the initial state. This yields a unique intermediate state to which the second action is applied etc. After all resulting states for all possible linearizations have been computed, the new initial state is obtained as the intersection of them. Note that in most cases, not all of the possible $n!$ linearizations need to be considered, because the n actions have to form a valid parallel plan (Koehler & Hoffmann 1998).

Planning incrementally for increasing subsets of goals from the agenda preserves completeness:

1. The plan π_1 is generated as a solution for the planning problem to reach $goal_1$ from the original initial state I.
2. Given π_1 and I, the new initial state for the second entry in the agenda is obtained as $R(I, \pi_1)$ with R performing the computations described above. Starting from this state, the planner generates the plan π_2 for the goal $goal_2$, which contains $goal_1$ as a subset.

n. In general, $R(I, \pi_1 \circ \pi_2 \circ \ldots \circ \pi_{n-1})$ yields the initial state for the planning problem $goal_n$ (containing the subsets of goals $goal_1 \ldots goal_{n-1}$) solved by plan π_n.

One could argue that planning for the increasing goal sets can lead to highly non-optimal plans. IPP, however, always tries to achieve goals "with no-ops first", see (Blum & Furst 1997). Since all goals $goal_1, goal_2, \ldots, goal_{n-1}$ do already hold in the initial state when the planner tries to achieve $goal_n$, this strategy ensures that these goals are only destroyed and reestablished if no solution can be found otherwise.

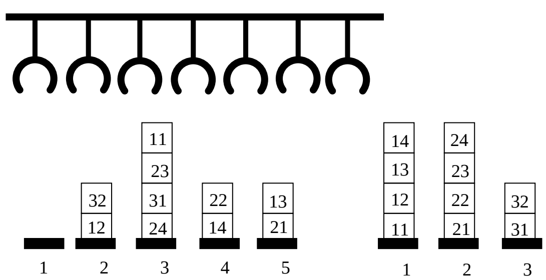

Figure 3: A blocksworld instance with limited space on the table, seven robot arms, and several stacks.

As an example, let us consider the following *blocksworld* problem taken from (El-Kholy & Richards 1996) where seven robot arms can be used to order 10 blocks into 3 stacks under space restrictions on the table, see Figure 3. The goal agenda derived by IPP orders the blocks into horizontal layers:

1. on-table(21,t2) $\wedge$ on-table(11,t1)
2. on-table(31,t3) $\wedge$ on(22,21) $\wedge$ on(12,11)
3. on(32,31) $\wedge$ on(13,12) $\wedge$ on(23,22)
4. on(14,13) $\wedge$ on(24,23)

The optimal plan of 20 actions solving the problem is found by IPP in 8 s, without goal analysis IPP needs approx. 12 minutes.[1]

[1] All times were measured on a Sun Ultra 1/170 running IPP 3.1 and are listed in seconds. By the time of publication, IPP 3.2 was released, which implemented significant changes to parts of the code concerning exclusivity information that were incorporated from the original Graphplan system. The changes led to dramatic improvements in runtime and memory consumption, e.g., IPP 3.2. solves this problem in 29 s. But since the goal agenda was implemented based on IPP 3.1 we list runtimes of IPP 3.1 for reasons of comparison and to illustrate the possible relative improvements.

Empirical Results

The current definition of $\prec$ that we used in this first implementation of the goal agenda works well in several domains such as *blocksworld*, *tyreworld*, *schedworld*, *briefcase*, *tower of Hanoi*, and an *office domain* variant. It is not adequate to order goals in the *logistics*, *trains*, or *rocket* domains, where no natural goal ordering seems to exist. We list a few examples to show how a goal agenda can significantly improve performance.

Figure 4 shows IPP on the SATPLAN examples from (Kautz & Selman 1996), the bw_large.e example taken from (Dimopoulos, Nebel, & Koehler 1997), and two very large examples bw_large.f and bw_large.g with 25 blocks/6 stacks and 30 blocks/8 stacks, resp.

SATPLAN	#actions	+G	+G+R	+G+R+L
bw_large.b	22	1.63	0.62	0.53
bw_large.c	48	24.71	2.91	2.92
bw_large.d	54	25.99	4.35	4.22
bw_large.e	52	26.01	3.99	3.97
bw_large.f	90	-	-	16.32
bw_large.g	84	-	321.91	29.33

Figure 4: IPP on the extended SATPLAN blocksworld test suite. The second column shows the plan length, +G means that IPP is using a goal agenda, +G+R means IPP uses goal agenda and RIFO, +G+R+L means that subgoals from the same set in the agenda are arbitrarily linearized. The times include also the effort spent on goal analysis and removal of irrelevants. A dash means that the system ran out of memory on a 1 Gbyte machine.

IPP without goal agenda can only solve bw_large.b for which it finds an optimal solution of 18 actions. Using a goal agenda, the plans become slightly longer (22 instead of 18 actions for the first example), because some blocks are accidentally put in positions where they cut off goals that are still ahead in the agenda, but performance is increasing dramatically. A further speed-up is possible when RIFO (Nebel, Dimopoulos, & Koehler 1997) is additionally used to remove irrelevant information from all subproblems, because this reduces the size of planning graphs dramatically. Finally, goals that belong to one set under $\prec$ and that occur in the same entry in the goal agenda can be solved in any order, i.e., +L means the planner arbitrarily linearizes these goals and with this option, the problems are solved almost instantly.

Figure 5 shows IPP on a simple stacking problem where n blocks are on the table in the initial state and the goal is a single stack of all blocks. IPP can handle up to 12 blocks in less than 5 minutes, but for 13 blocks more than 15 minutes are needed. Using a goal agenda, 40 blocks can be stacked in less than 5 minutes. With the goal agenda and RIFO, the 5 minutes limit is extended to 80 blocks and the 100 blocks stacking problem is solved in 11.5 minutes including 11.3 minutes for goal analysis and removal of irrelevants. It is interesting to analyze how the total problem solving time is shared, see (Koehler & Hoffmann 1998). The time to determine the goal agenda takes between 50 and 60 per cent, the removal of irrelevants from each subproblem takes between 40 and 50 per cent and the actual planning time for all subproblems is reduced to approx. 1 %. This indicates that a further speed-up is possible when improving the time for the two analysis procedures. It also indicates that even the hardest planning problems can become easy if they are structured and decomposed in the right way.

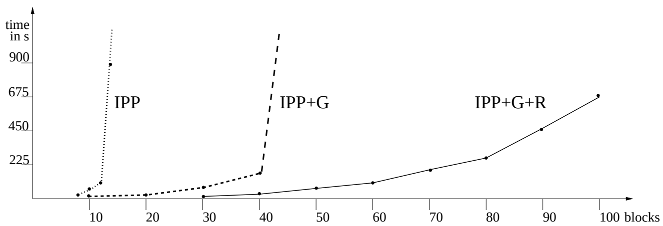

Figure 5: IPP on a simple, but huge stacking problem.

Figure 6 shows IPP on the fixit problem with an increasing number of flat tires that need to be changed. Solution length is slightly increasing which is caused by superfluous jack-up and jack-down actions because wheels are mounted on the hubs, but do not need to be immediately secured with nuts. However, to add the nuts, a hub must be jacked-up. Adapting the operators to the problem of fixing several tires, the optimal plans are generated.

Tires	IPP	+G+R	+G+R+L
1	0.10 (12/19)	0.15 (14/19)	0.16 (17/19)
2	17.47 (18/30)	0.44 (24/32)	0.35 (30/34)
3	-	6.15 (33/45)	0.68 (41/48)
4	-	-	1.12 (52/60)
5	-	-	2.04 (63/73)
10	-	-	16.89 (118/136)

Figure 6: IPP in the Tyreworld. Numbers in braces list the time steps and number of actions in the generated plan.

In the case of 3 tires, the following goal agenda is computed (only listing the new goals in each entry):

1: inflated(r3) $\wedge$ inflated(r2) $\wedge$ inflated(r1)

2: on(r3,hub3) $\wedge$ on(r1,hub1) $\wedge$ on(r2,hub2)

3: tight(n2,hub2) $\wedge$ tight(n3,hub3) $\wedge$ tight(n1,hub1)

4: in(w3,boot) $\wedge$ in(pump,boot) $\wedge$ in(w1,boot) $\wedge$ in(w2,boot)

5: in(jack,boot)

6: in(wrench,boot)

7: closed(boot)

The hardest subproblem in the agenda is to achieve the *on*-goals, i.e., to mount inflated spare wheels on the various hubs. Trying to generate a maximum parallelized plan is impossible for IPP for more than 3 tires. But since the goals are completely independent of each other, using any linearization of them will perfectly work. In the case of 10 tires only 2662 actions are tried to generate a plan of 136 actions within 0.08 s. It took 0.55 seconds to generate the goal agenda, 14.42 s to remove irrelevant information from all subproblems, 1.74 s to generate the various planning graphs, and 0.10 s to compute the initial states for all subproblems.

Applying a random linearization to some or all goal sets in the agenda can be considered as another kind of a *domain-independent* "definition" of $\prec$, i.e., one can apply one possible relation definition to come up with an initial agenda where the various entries are further ordered using other relations. Note that randomly ordering goals from the same set is also justified by the agenda computation based on $\prec$. We assign nodes with the same degree in the transitive closure of the graph to the same set to improve the optimality of the solution plans, but one could also use any arbitrary topological ordering over nodes having the same degree.

Intermediate Goals

A solution to a subproblem from the agenda is constructed independently of what planning problems are still ahead. This can lead to non-optimal plans. To improve plan quality, we introduce a refinement of the goal agenda with *intermediate goals*.

An *intermediate* goal is a goal that the planner *necessarily* has to make true before it can achieve an original goal g *independently* of the action that will be selected to achieve g. Examples are *clear* or *holding* goals in the blocksworld, or *in(?object)* in the briefcase domain where objects have to be put into the briefcase before they can be moved.

Intermediate goals can be seen as "safe spots" when searching the state space, in the sense that **every** plan solving the goal has to temporarily achieve the intermediate goal. Information about intermediate goals can be used in two ways:

1. To avoid that a goal ahead in the goal agenda is cut-off when planning for an earlier goal entry.
2. To further refine entries in the goal agenda that form subproblems which are still too complex for the planner.

Given an entry from the agenda, we have developed a backchaining approach similar to the one described in (Nebel, Dimopoulos, & Koehler 1997) to derive intermediate goals. Due to space restrictions, we can only give a very brief sketch of the main ideas and need to refer the interested reader to (Koehler & Hoffmann 1998). Given an atomic goal, all actions whose effects match the goal are considered. For each action, the required preconditions and effect conditions are determined and the intersection over the various condition sets forms the intermediate goals. For a set of atomic goals, the union over the individual intermediate goals is formed to obtain the intermediate goals for the goal set. Two possible problems occur:

1. The set of intermediate goals can become unsolvable. For example, the conjunctive goal *holding(a)* and *holding(c)* is unsolvable if only one robot arm is available.
2. Intermediate goals derived for an entry at position k in the agenda could in principal be forwarded to all entries at positions earlier than k, but this again can lead to unsolvable problems.

There are no general solutions to this problem. If a set of intermediate goals is unsolvable, one can either abandon it, or avoid the computation of the union over intermediate goals for different subgoals by randomly ordering the subgoals. Currently, we restrict the possible ways how intermediate goals are added to the agenda and use them to augment the entry in the agenda that immediately precedes the entry for which they were generated, e.g., if the agenda was [1:a, 2:b] and c is an intermediate goal for b then the agenda is refined into [1:a $\wedge$ c, 2:b]. If the subproblem $a \wedge c$ is unsolvable, the planner returns to the original agenda.

Figure 7 shows the performance of IPP generating the optimal plan using a goal agenda refined with intermediate goals for the roundtrip problem from the briefcase domain where an increasing number of objects at different locations needs to be taken home.

#objects	IPP	+G+I+R	+G+I+R+L
4	1.55	0.58	0.12
5	100.86	40.10	0.19
6	-	3816.49	0.30
8	-	-	0.51
10	-	-	0.90

Figure 7: IPP on briefcase roundtrip problems using a goal agenda with intermediate goals (+G+I), RIFO (+R), and linearization (+L).

Conclusion and Outlook

The paper introduces an approach to order sets of conjunctive goals. Based on the definition of a relation $\prec$ between atomic goals, a goal agenda of totally ordered and increasing sets of subgoals is computed, which causes only polynomial overhead for the planner, but can lead to an exponential efficiency gain.

Experimenting with other domain-independent or domain-dependent definitions of $\prec$ is one possible future direction for research. Furthermore, the combination of various definitions of $\prec$ and their potential to speed up planners should be worth exploring.

References

Barrett, A., and Weld, D. 1993. Characterizing subgoal interactions for planning. In *IJCAI-93*, 1388–1393.

Blum, A., and Furst, M. 1997. Fast planning through planning graph analysis. *AIJ* 90(1–2):279–298.

Chapman, D. 1987. Planning for conjunctive goals. *AIJ* 32(3):333–377.

Cheng, J., and Irani, K. 1989. Ordering problem subgoals. In *IJCAI-89*, 931–936.

Dimopoulos, Y.; Nebel, B.; and Koehler, J. 1997. Encoding planning problems in non-monotonic logic programs. In *ECP-97*, 169–181.

Drummond, M., and Currie, K. 1989. Goal ordering in partially ordered plans. In *IJCAI-89*, 960–965.

El-Kholy, A., and Richards, B. 1996. Temporal and resource reasoning in planning: the parcPLAN approch. In *ECAI-96*, 614–618.

Gerevini, A., and Schubert, L. 1996. Accelerating partial-order planners: Some techniques for effective search control and pruning. *JAIR* 5:95–137.

Hertzberg, J., and Horz, A. 1989. Towards a theory of conflict detection and resolution in nonlinear plans. In *IJCAI-89*, 937–942.

Irani, K., and Cheng, J. 1987. Subgoal ordering and goal augmentation for heuristic problem solving. In *IJCAI-87*, 1018–1024.

Joslin, D., and Roach, J. 1990. A theoretical analysis of conjunctive-goal problems. *AIJ* 41:97–106.

Kautz, H., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *AAAI-96*, 1194–1201.

Koehler, J., and Hoffmann, J. 1998. Planning with goal agendas. Technical report, University of Freiburg. http://www.informatik.uni-freiburg.de/~koehler/ipp.html.

Koehler, J.; Nebel, B.; Hoffmann, J.; and Dimopoulos, Y. 1997. Extending planning graphs to an ADL subset. In *ECP-97*, 273–285.

Koehler, J. 1996. Planning from second principles. *AIJ* 87(1-2):148–187.

Korf, R. 1987. Planning as search: A quantitative approach. *AIJ* 33:65–88.

Nebel, B.; Dimopoulos, Y.; and Koehler, J. 1997. Ignoring irrelevant facts and operators in plan generation. In *ECP-97*, 338–350.

Pollack, M.; Joslin, D.; and Paolucci, M. 1997. Selection strategies for partial-order planning. *JAIR* 6:223–262.