Planning with Goal Agendas

Jana Koehler Jörg Hoffmann

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

July 1998

TECHNICAL REPORT No. 110

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.

This technical report is an extended and updated version of the paper Solving Complex Planning Tasks Through Extraction of Subproblems *that has been published at AIPS-98.*

Contents

1 Introduction

How to effectively plan for interdependent subgoals has been in the focus of AI planning research for a very long time [Cha87]. 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., [DC89, HH89]. On the other hand, one can perform a structural analysis of the planning task to determine an appropriate goal ordering before planning starts, see [IC87, CI89, JR90].

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 [JR90] or severe restrictions need to be put on the representation [CI89].

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 [Kor87, BW93] 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 study two possibilities. First, an approach of *incrementally* planning for the increasing goal subsets from the agenda is proposed. This way, the efficiency of planning can be dramatically improved in many cases while preserving the completeness of the planning algorithm. Second, we outline how information from the goal agenda can be used *during* planning by ordering goal sets that are generated by the search process.

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.

2 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 $\mathrm{eff}_i^-(o)$. The set of all effects is $\Phi(o) = \bigcup_{i=0}^{n} \phi_i(o)$.

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 $A = \mathit{paint}(x)$ and $B = \mathit{polish}(y)$ 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 over $\mathcal{O}$ exists that*

1. *is executable in some state in which A holds, but B does not hold,*
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 is 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.

2.1 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 [BF95]. Starting from I with $\mathcal{O}$, the planning graph is built until it reaches its *fix point*. All atoms, which are marked as exclusive of A in the fix point 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.

$$\mathsf{F} = \{f \mid f \text{ is exclusive of } A \text{ in the fix point level}\}$$

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).

In order to generate planning graphs for larger planning problems one needs to develop a sophisticated algorithm that does not build the graph explicitly to reduce cpu time and memory consumption for this analysis phase. Alternatively, one can analyze the available actions directly for which we have developed a method called *direct operator analysis*. It works as follows: $\mathsf{F}(A, o)$ contains all facts that are always made false when A is made true using an action o. This means for a single action that achieves A unconditionally that its unconditional negative effect $\text{eff}_0^-(o)$ would belong to F. If it achieves A with a conditional effect ϕ_i, then its unconditional negative effect $\text{eff}_0^-(o)$, the conditional negative effect $\text{eff}_i^-(o)$, and all conditional negative effects $\text{eff}_j^-(o)$ 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.

$$\mathsf{F}(A) = \bigcap_{\{\phi_i(o) \in \Phi(o) \mid o \in \mathcal{O}, A \in \text{eff}^+(o)\}} F(A, o)$$

$$F(A, o) = \bigcup_{\substack{\{\text{eff}_j^-(o) \in \Phi(o) \mid pre_j(o) \subseteq pre_i(o)\} \\ A \in \text{eff}_i^+(o)}} \text{eff}_j^-(o)$$

Figure 1 compares both analysis methods for some larger examples. The planning graph algorithm does not explicitly build the graph, but the representation of exclusivity information is still quite memory consuming and can probably still be improved.

problem	fix-point	time in s	memory(graph/exclusives) in Kb	DOI time in s
manhattan	52	94.3	679/66,413	3.83
hanoi8	11	0.98	188/2,293	0.25
woodshop2	4	0.02	21/200	0
montlake	11	0.04	22/573	0
schedworld6	9	0.45	131/2,847	0.12
briefcase5	7	0.17	80/918	0.04
logistic.c	11	0.90	182/3071	0.30
sokoban2	47	8.98	247/21,890	0.44

Figure 1: Effort to determine F with planning graphs compared to the time needed when direct operator analysis (DOI) is used.

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. Depending on the method how to compute F different goal agendas can result in some domains. Which method to choose is subject to heuristic experimentation since no general conclusions could be drawn from our evaluation.
As an example, let us consider the *blocksworld* domain with the planning problem

initial: *on(a,b) on(b,c) on(c,d) on(d,table) clear(a) clear(table)*
goal: *on(a,table) on(b,a) on(c,b) on(d,c)*

and only one operator that puts a block x from a block z on top of a new block y:

puton*(?x ?y ?z: block)*
on(?x,?z) clear(?x) clear(?y) $?x \neq table$
ϕ_0: *on(?x,?y)* $\neg$ *on(?x,?z)*
ϕ_1: $?z \neq table \Rightarrow$ *clear(?z)*
ϕ_2: $?y \neq table \Rightarrow \neg$ *clear(?y)*

The set of instantiated **puton** actions comprises only STRIPS operators, because the inequality condition is evaluated during instantiation and only valid instances are generated. For example, we obtain

puton*(c,d,table)*
on(c,table) clear(c) clear(d)
ϕ_0: *on(c,d)* $\neg$ *clear(d)* $\neg$ *on(c,table)*

The resulting F sets that are generated by the two methods for each atomic goal are shown in Figure 2. As said above, planning graphs generate larger F sets in this STRIPS domain, but these do not necessarily yield better goal agendas. In the case of conditional actions, an inverse picture is obtained as Figure 3 shows for the *briefcase* domain with the following **move** operator

move(*?l: location*)
ϕ_0: *is-at(?l)*
ϕ_1: $\forall$ *?m location ?m* $\neq$ *?l* $\Rightarrow \neg$ *is-at(?m)*
ϕ_2: $\forall$ *?o object ?m location ?m* $\neq$ *?l in(?o)* $\Rightarrow$ *at(?o ?l)* $\neg$ *at(?o ?m)*

and the planning problem

initial: *is-at(home) at(o1 l1) at(o2 l2) at(o3 l3) at(o4 l4)*
goal: *is-at(home) at(o1 home) at(o2 home) at(o3 home) at(o4 home)*

goal	planning graphs	direct operator analysis
on(d,c)	clear(c) on(d,table) on(c,d) on(b,c) on(a,c) on(d,a) on(d,b)	clear(c)
on(c,b)	clear(b) on(c,d) on(b,c) on(a,b) on(d,b) on(c,a) on(c,table)	clear(b)
on(b,a)	clear(a) on(b,table) on(b,c) on(a,b) on(d,a) on(c,a) on(b,d)	clear(a)
on(a,table)	on(a,b) on(a,c) on(a,d)	$\emptyset$

Figure 2: Comparison of both methods in a STRIPS domain

goal	planning graphs	direct operator analysis
at(o4,home)	$\emptyset$	at(o4,l4) at(o4,l3) at(o4,l2) at(o4,l1) is-at(l4) is-at(l3) is-at(l2) is-at(l1)
at(o3,home)	$\emptyset$	at(o3,l4) at(o3,l3) at(o3,l2) at(o3,l1) is-at(l4) is-at(l3) is-at(l2) is-at(l1)
at(o2,home)	$\emptyset$	at(o2,l4) at(o2,l3) at(o2,l2) at(o2,l1) is-at(l4) is-at(l3) is-at(l2) is-at(l1)
at(o1,home)	$\emptyset$	at(o1,l4) at(o1,l3) at(o1,l2) at(o1,l1)
is-at(home)	is-at(l1) is-at(l2) is-at(l3) is-at(l4)	is-at(l1) is-at(l2) is-at(l3) is-at(l4)

Figure 3: Comparison of both methods in an ADL domain

As a side-effect of the DOI analysis method, one could use the information from generated F sets to add additional exclusivity information to planning graphs. For example, in the above*briefcase* problem, on cannot derive that the briefcase and each object can only be in one location at any time, i.e., facts such as *at(o4,l4)* and *at(o4,l3)* are not exclusive. But using F sets, this information could be added to the graph, which should reduce the search space for the planner in many cases.[1]
In the next step, the set F is now used to exclude some actions from the set $\mathcal{O}$.

[1]Unfortunately, no search space reduction is obtained in the *briefcase* domain because such subgoals are never generated during planning.

2.2 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 or $\neg A$ 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.

$$\mathcal{O}^R = \{red(o, \mathsf{F}, A) \mid o \in \mathcal{O}, A \notin \text{eff}_0^-(o), \text{ and } \text{pre}_0(o) \cap \mathsf{F} = \emptyset, \neg A \notin pre_0(o)\}$$

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

$$\begin{aligned} \text{pre}_0(o') &= \text{pre}_0(o) \\ \Phi_i(o') &= \Phi(o) \setminus \{\phi_i(o) \mid \text{pre}_i(o) \cap \mathsf{F} \neq \emptyset \text{ or } A \in \text{eff}_i^-(o)\} \end{aligned}$$

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 fix point 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 fix point 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 fix point computation, we test if there exists an atom $f \in \mathsf{F}$ with $f \in \text{eff}_i^+(o)$ where $o \in \mathcal{O}^R$. If this is the case, we determine $\mathsf{F} = \mathsf{F} \setminus \{f\}$ 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.

Returning to the previous the *blocksworld* example and the problem whether the atomic goal $on(b, a)$ can be reached after $on(c, b)$ has been achieved, using planning graphs the goal agenda manager computes $\mathsf{F}_{on(c,b)} = \{clear(b), on(c, d), on(b, c), on(a, b), on(d, b)$ $on(c, a), on(c, table)\}$, which excludes 36 from the 48 ground actions such that only 12 remain which can manipulate the remaining blocks d and a in a limited way. In particular, all actions that achieve $on(b, a)$ are eliminated because they require $clear(b)$ as precondition. With direct operator analysis, we obtain with $\mathsf{F}_{on(c,b)} = \{clear(b)\}$ a much smaller set, but this also excludes all actions that could achieve the goal $on(b, a)$.

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 reduced actions $\mathcal{O}^R$, the relation $B \prec_2 A$ holds if and only if no plan over $\mathcal{O}^R$ exists that*

1. *is executable in some state in which A holds, but B does not hold,*
2. *achieves B.*

Obviously, relation $\prec_2$ implies $\prec_1$. 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.

2.3 Deriving an Approximative Test

Definition 2 formulates a test of the abstract structure: $\neg\,\exists plan : exec(plan, s_{A,}) \wedge achieves(B)$, which is equivalent to $\forall plan : achieves(B) \rightarrow \neg exec(plan, s_A)$, i.e., for all plans has to be shown that if they achieve B then they are not executable in the state satisfying A, but not B. The condition $s_A \models A$ has already been reflected in the reduction operation on actions, but $s_A \not\models B$ has not been considered so far. In fact, one could also eliminate all actions and conditional effects that require B from the reduced action set, but this would require to compute different F set for each pair A, B one is interested in. A more simple, but very powerful heuristic is to assume that B does not hold in s_A.

To implement the test $\forall p : achieves(B) \rightarrow \neg exec(p, s_A)$, sufficient and easy-to-test conditions are checked. First, it is tested that no action in $\mathcal{O}^R$ has B as a positive effect. In this case, no plan can achieve B under the assumption that B does not yet hold. If the first test fails, it is shown secondly that each action that has B as a positive effect, requires 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*

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

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 $\prec_1$.

The following simple example illustrates the computation of $\prec$ relations based on Definition 3. It uses a common representational variant of the *blocksworld* using four

possible actions to stack, unstack, pickup and putdown blocks. Given the two goals *on(a,b)* and *on(b,c)* (note that we don not specify where the block c has to go, but leaves this to the planner) together with the following two actions that can achieve these goals, is it still possible to stack block b on c with a on top of b assuming that *on(a,b)* has been achieved?

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).*

Using planning graphs yields $\mathsf{F} = \{holding(a), clear(b), holding(b), on-table(a), \dots\}$ 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 $\mathcal{O}^R$.

3 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.

3.1 Transitivity of the $\prec$ Relation

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 immediately be extended to increasing sets. If the planner plans from A to B and both atoms are goals in a solvable planning problem, then it must be possible to plan from A to $A \wedge B$ etc. This yields the graph nodes shown in Figure 4. 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 can be found in time less than cubic in the number of nodes in the graph.

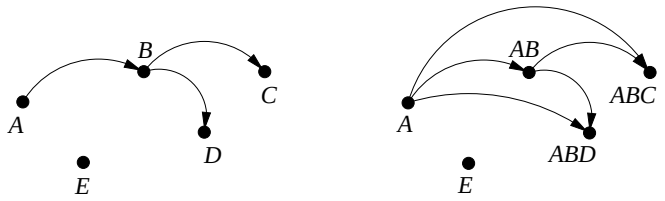

Figure 4: On the left, the original $\prec$ relations as computed during the analysis. On the right, the transitive closure of the graph extended to increasing goal sets.

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, see Figure 5. 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.

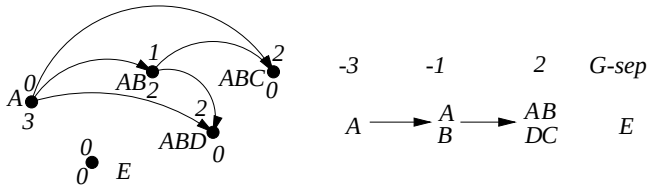

Figure 5: On the left, the in- and outdegrees of the nodes are shown. The node E becomes a member of the *G-sep* set, while the remaining nodes are assigned to three increasing and totally ordered sets based on their degree. With an empty *G-sep* the goal agenda were $[1 : A, 2 : A \wedge B, 3 : A \wedge B \wedge C \wedge D]$.

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 can successfully be 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.

3.2 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 $\{A_1, \ldots, A_n\}$. The extension of exclusivity from pairs of atoms to arbitrary sets of atoms is straightforward, i.e., an atom is exclusive of a set of atoms if it is exclusive of at least one atom from the set.

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

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

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 *some* the goal 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 some 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 accordingly:

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 6.

Definition 4 defines a graph with the various sets as vertexes and the $\prec_S$ relation as edges. For this graph, the nodes are extended to increasing sets and the transitive closure is determined yielding to a degree for each node. If it yields a total ordering over all nodes, the goal agenda contains an entry for each node (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 seems to make not much sense. A collapse of the goal agenda can be interpreted as a failure of the definition for $\prec$ to characterize constraints between goals. One should therefore use a different domain-independent or domain-dependent relation.

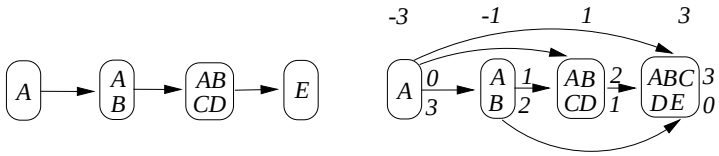

Figure 6: On the left, a directed graph based on the sets and *G-sep* showing the existence of a relation $\{A, B, C, D\} \prec_S \{E\}$. Note that a goal analysis between singleton sets does not need to be repeated. On the right, the transitive closure showing in- and outdegrees and the successfully computed goal agenda for the example based on the totally ordered sets.

4 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 [GS96] or LCFR [PJP97]. Second, one can imagine to reuse plans for subproblems stated by the agenda and plan from second principles [Koe96]. 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 in more detail and will shortly sketch the impact of goal agenda information during planning at the end of this section.

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 [KNHD97] 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.

4.1 Computing Initial States for Subproblems

The computation of the new initial state follows exactly the semantics of parallel ADL plans as defined for IPP [KNHD97]. We define a function R that given a unique initial state and a set of parallel actions with conditional effects, generates one possible linearization. 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.

The result of applying a single ADL action to a uniquely defined state S is obtained with

$$Res(S, o) := \begin{cases} S \cup A(S, o) \setminus D(S, o) & \text{pre}_0(o) \subseteq S \\ \text{undefined} & \text{otherwise} \end{cases}$$

where $A(S,o)$ and $D(S,o)$ are defined as

$$\begin{aligned} A(S,o) = &\quad \bigcup_i \mathrm{eff}_i^+(o) \text{ with } \mathrm{pre}_i(o) \subseteq S \\ D(S,o) = &\quad \bigcup_i \mathrm{eff}_i^-(o) \text{ with } \mathrm{pre}_i(o) \subseteq S \end{aligned}$$

Given a parallel set of actions $O = \{o_1, \ldots, o_n\}$, the set of atoms that holds in all resulting states is determined with ($Seq(O)$ denotes all possible linearizations of O)

$$R(S,O) = \begin{cases} \bigcap\{T | q \in Seq(O), T = Res(S,q)\} & Res(S,q) \text{ defined } \forall q \in Seq(O) \\ \text{undefined} & \text{otherwise} \end{cases}$$

For a valid IPP plan as a sequence of parallel action sets $\langle O_1, \ldots, O_n \rangle$ we define

$$R(S, \langle O_1, \ldots, O_n \rangle) = R(R(S, \langle O_1 \ldots O_{n-1} \rangle), \langle O_n \rangle)$$

Note that in most cases, not all of the possible $n!$ linearizations of a parallel action set containing n actions need to be considered, because the n actions have to form a valid parallel plan, i.e., all possible conflicts between conditional effects have been resolved. In particular, if only actions with unconditional effects are contained in the set, just one arbitrary linearization is considered, because the resulting state is unique. The experiments showed that determining the new initial state takes almost no time and is neglectible compared to the effort spent on building planning graphs, see Section 5. This is mainly caused by the fact that the plans which are generated for the common planning problems usually do not contain much parallelism, i.e., more than 5 actions in parallel is extremely rare.

4.2 Preserving Completeness

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 defined 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, $R(I, \pi_1 \circ \pi_2 \circ \cdots \circ \pi_{n-1})$ 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, is based on planning graphs, which were originally introduced by the GRAPHPLAN system [BF95], and always tries to achieve goals "with no-ops first". Since all goals $goal_1, goal_2, \ldots, goal_{n-1}$ are already achieved in the initial state when the planner tries to achieve $goal_n$, this strategy ensures that these goals are only destroyed and re-established if no solution can be found otherwise.

As an example, let us consider the following *blocksworld* problem taken from the Parcplan system [EKR96] where seven robot arms can be used to order 10 blocks into 3 stacks under space restrictions on the table, see Figure 7. The goal agenda derived by IPP orders the blocks into horizontal layers:

1: on-table(21,t2) ∧ on-table(11,t1)
2: on-table(31,t3) ∧ on(22,21) ∧ on(12,11)
3: on(32,31) ∧ on(13,12) ∧ on(23,22)
4: on(14,13) ∧ on(24,23)

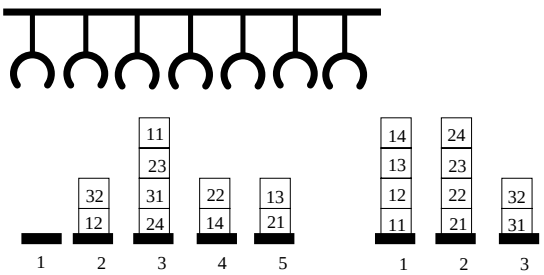

Figure 7: A blocksworld instance with limited space on the table, seven robot arms, and several stacks.

The optimal plan of 20 actions solving the problem is found by IPP in 14 s, where it spends 1 second on computing the goal agenda, almost 13 seconds to build the planning graphs, but only 0.01 second to search for a plan. Only 70 actions have to be tried to find the solution. Without goal analysis IPP needs approx. 47 s and searches 52893 actions in more than 26 seconds.[2] Using RIFO [DNK97] to remove irrelevants does not help in solving the planning problem as a whole, but helps when planning for the goal sets from the agenda. It reduces runtime down to less than 8 s with 1 s again spent on the goal agenda, almost 6 seconds spent on the removal of irrelevant actions and initial facts, less than 1 s spent on building the planning graphs, and as previously almost no time spent on planning.

4.3 Goal Agenda Information to Guide Search

It is well-known that goal ordering information can significantly reduce search effort. This observation also carries over to planners based on planning graphs. The *blocksworld* example above illustrates this effect very nicely where the planner is fed with the following goal state description:

on-table(11 t1) on-table(21 t2) on-table(31 t3) on(12 11) on(13 12)
on(14 13) on(24 23) on(23 22) on(22 21) on(32 31)

In order to find a plan for this problem, as already said 52893 actions have to be tried, which takes more than 26 s. But feeding the (optimal) goal ordering that results from the agenda into the planner

on-table(21,t2) on-table(11,t1) on-table(31,t3) on(22,21)
on(12,11) on(32,31) on(13,12) on(23,22) on(14,13) on(24,23)

[2]All times were measured on a Sun Ultra 1/170 under Solaris running IPP 3.3 and are listed in seconds. The implemented code and all example domains can be downloaded from the IPP homepage at http:www.informatik.uni-freiburg.de/~ koehler/ipp.html.

reduces the search space down to 5977 actions, which corresponds to only 0.86 s search effort and only 22 seconds of total planning time. But trying the planner on the reversed optimal (worst) ordering increases the search space to 3759616 actions that have to be tried until a plan is found, which corresponds to almost half an hour of total planning time.

The simple experiment shows that using the goal agenda can also serve to effectively order the goals when addressing a planning problem as a whole. One can even go further and always order goal sets when they are generated during search. Here, one can limit the goal analysis to the set of actions that is contained in the subgraph spanned by the current goal sets, because only those will be considered by the planner to achieve the goals. Besides this, one can limit the goal set only to those actions that can be used to achieve the goal set at the current level in the graph and order the goals based on this *local* information.[3] A deeper analysis of such *on-line* goal agendas goes beyond the scope of this paper and also requires further investigation.

5 Empirical Results

The current definition of $\prec$ that we used in this first implementation of the goal agenda approach works well in several domains such as *blocksworld*, *tyreworld*, *schedworld*, *woodshop*, *briefcase*, *tower of Hanoi*, and an *office domain* variant. This domains share the characteristic that usually only one optimal solution exists and that there is a natural order in which goals have to be addressed in order to reach this optimal solution.

It is not adequate to order the goal sets that are commonly considered in the *logistics*, *trains*, or *rocket* domains, which seems to be caused by the phenomenon that in these domains many different possible shortest plans are correct solutions.[4] Furthermore, it is obvious that goal agendas cannot improve planning in domains where the goal is just given by a single atom (or a small goal set) such as *manhattan* world, *sokoban* or *molgen*.

5.1 Blocksworld

We list a few examples to show how a goal agenda can significantly improve performance. Figure 8 shows IPP on the SATPLAN examples from [KS96], the bw_large.e example taken from [DNK97], and two very large examples bw_large.f and bw_large.g with 25 blocks/6 stacks and 30 blocks/8 stacks, resp.

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 [NDK97] 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

[3] According to personal communication with D. Long and M. Fox such a technique has been used by the STAN planner in the first planning systems competition, see http://www.dur.ac.uk/~dcs0www/research/stanstuff/stanpage.html for upcoming details.

[4] But it could still be possible that on-line generated goal sets can be ordered.

SATPLAN	#actions	+G	+G+R	+G+R+L
bw_large.a	11	0.74	0.58	0.34
bw_large.b	22	0.86	0.55	0.52
bw_large.c	48	7.34	2.42	2.58
bw_large.d	54	11.62	3.74	3.81
bw_large.e	52	11.14	3.99	3.97
bw_large.f	90	-	-	16.01
bw_large.g	84	-	117.56	28.71

Figure 8: 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.

planner arbitrarily linearizes these goals and with this option, the problems are solved almost instantly.

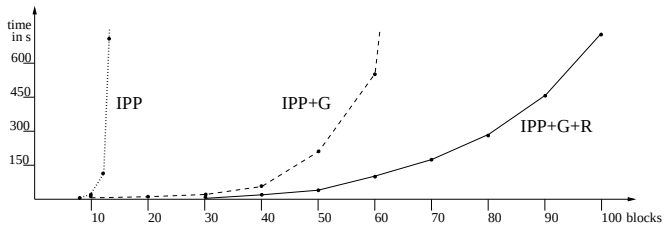

Figure 9: IPP on a simple, but huge stacking problem.

Figure 9 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.

problem	goal agenda	RIFO	planning
stack 20	0.31 = 16 %	1.44 = 75 %	0.13 = 7 %
stack 40	1.57 = 7 %	18.77 = 90 %	0.51 = 2 %
stack 60	4.40 = 4 %	93.10 = 94 %	1.15 = 1 %
stack 80	9.60 = 3 %	283.60 = 96 %	2.33 = 1 %
parcplan	0.86 = 12 %	5.52 = 76 %	0.83 = 11 %

Figure 10: Sharing of problem solving on blocksworld examples between GA, RIFO, and IPP's planning time, which includes the time to build and search the planning graph.

The time to determine the goal agenda takes between 3 and 16 per cent, the removal

of irrelevants from each subproblem takes between 75 and 96 per cent and the actual planning time for all subproblems can be reduced down to approx. 1 %. The overall problem solving time is clearly determined by the time to remove irrelevant information from all subproblems that where extracted by the goal analysis. The actual planning time becomes a marginal factor in the determination of performance. 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.

5.2 Tyreworld

Figure 11 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 secured immediately with nuts. However, to add the nuts, a hub must be jacked-up.

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)	0.41 (24/32)	0.32 (30/34)
3	-	2.87 (32/44)	0.63 (41/48)
4	-	-	1.12 (52/60)
5	-	-	1.93 (63/73)
10	-	-	16.72 (118/136)

Figure 11: 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.

5.3 Tower of Hanoi

A surprising result was obtained for *tower of hanoi* problems. Although a perfect agenda is returned that describes the right ordering in which the discs have to be stacked, IPP was not able to benefit from this information. The goal agenda leads to a partition into subproblems that corresponds to the recursive formulation of the problem solving algorithms, i.e., to solve the problem for n discs, we first have to solve the problem for $n-1$ discs, etc. In the case of three discs $d1$, $d2$, $d3$ of increasing size, the following agenda is returned

1: on(d3,stack3)
2: on(d2,d3)
3: on(d1,d2)

For the first entry, a plan of 4 actions (time step 0 to 3) is generated which solves the problem for one disc. Then, for the second and first entry, a plan of 2 actions (time step 4 and 5) solves the problem for 2 discs where the first disc is already in the goal position. Finally, a one-step plan (time step 6) is generated for the third entry and solves the problem for n discs, with $n-1$ discs being already in the goal position. The runtimes of IPP, however, increase dramatically as Figure 12 shows.

```
time step 0:  move(d1,d2,stack3)
time step 1:  move(d2,d3,stack2)
time step 2:  move(d1,stack3,d2)
time step 3:  move(d3,stack1,stack3)
time step 4:  move(d1,d2,stack1)
time step 5:  move(d2,stack2,d3)
time step 6:  move(d1,stack1,d2)
```

This phenomenon is explained by the huge increase in the search space that is caused by the division into subproblems as it is reflected in the completely changed results for subset memoization. For example, for the problem of 6 discs IPP memoizes 32418 unsolvable subsets and can successfully match 220 goal sets to them. Using the goal agenda, 902081 goal sets are generated and memoized, obviously leading to a much higher branching in the search space. Of these, 69533 sets can be matched i.e., each 13th generated set does not cause any additional search, but still 832548 goal sets have to be searched compared to 32,198 sets in the case that no agenda is used. For the problem of 7 discs, 18.3 million goal sets are tried in contrast to 2.25 million goal sets without the agenda.

discs	IPP	IPP + G
3	0.08	0.07
4	0.33	0.25
5	1.57	3.10
6	9.71	88.45
7	69.44	2339.94

Figure 12: Runtimes of IPP with and without goal agenda on *hanoi* problems.

5.4 Woodshop and Scheduling World

In both domains, objects can be given different shapes, surfaces and colors, holes can be drilled etc. Shaping and drilling usually destroys surfaces including colors and makes objects dusty. Drilling requires that objects be in certain shapes, but shaping and painting require that objects obey certain surface and temperature conditions. Two typical examples are listed in the Appendix.[5] Using goal agendas plus RIFO, planning time decreases for the schedworld problem from 3.86 s down to 1.63 s, but at the price of optimality. Interestingly, removal of irrelevants can preserve optimality of the solution in some cases. For the tiny woodshop problem runtime is increasing from 0.08 s to 0.12 s, but for larger problems, significant improvements are obtained, i.e., previously unsolvable problems are solved in a few seconds, in particular if linearization of goal agendas is activated.

6 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 [NDK97] to derive intermediate goals. 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

[5]The problems can be be downloaded from the IPP homepage.

condition sets forms the intermediate goals. Figure 13 shows the computation of intermediate goals for the goal atom *on(a,b)*, which can only be achieved by the single action **stack(a,b)**. Consequently, the preconditions of this action *clear(b)* and *holding(a)* form the intermediate goals at level 1. With these goals, the backchaining process is called recursively. The goal *clear(b)* can be achieved with all actions that unstack something from *b*, the action *putdown(b)*, or any action that stacks *b* onto something. Each of the three action kinds have one common precondition, but there is no precondition that all these *unstack*, *putdown*, and *stack* have in common. Therefore, the intersection of their preconditions is empty and the backchaining process terminates with the last non-empty intermediate goal that was found. The goal *holding(a)* can either be achieved by *pickup(a)* or any *unstack(a,?x)* action. All these actions require the atoms *clear(a)* and *arm-empty* as preconditions, which form the intermediate goals at level 2. Trying to further backchain from the goals leads to an empty intersection over the preconditions. Thus, the planner returns the lowest framed preconditions from the figure as intermediate goals.

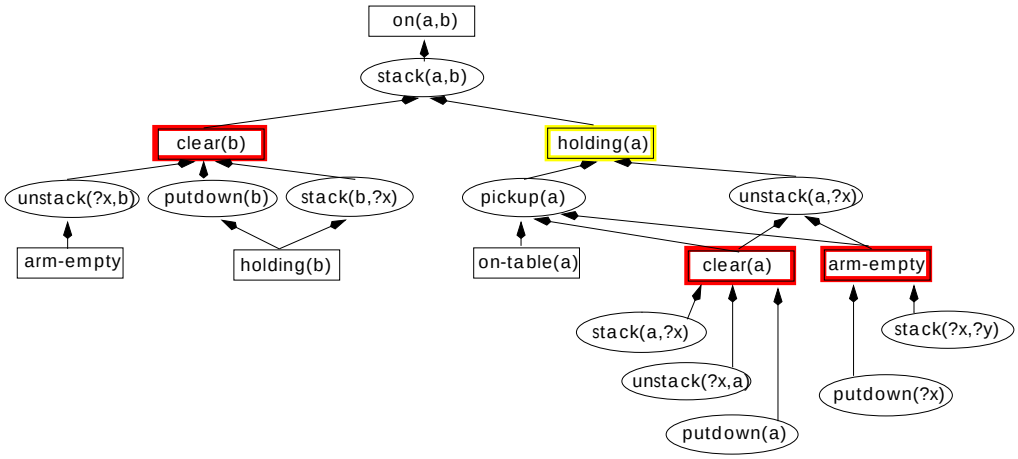

Figure 13: Computation of Intermediate goals for an Atomic Goal in the Blocksworld

In general, there is no universal method that can decide to which depth the backchaining process should be executed. Currently, no depth limit is imposed, but backchaining continues until either the intersections of preconditions become empty or a cycle occurs. The backchaining process can loop in domains where operators with symmetric preconditions and effects occur, i.e., one needs to explicitly test for re-occurring sets of intermediates in order to guarantee termination. Figure 14 shows an example for the *briefcase* domain.

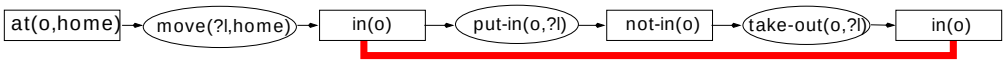

Figure 14: Cycles during the Computation of Intermediate Goals

The results of such a backchaining process can be used to refine the agenda. For example, before the planner addresses the goal *on(a,b)* it tries to reach *clear(b)* and *holding(a)*. When working on *holding(a)* it tries to reach *clear(a)* and *arm-empty* first. In some domains, when no ordering of the originally given goals is possible, intermediate goals help to come up with any agenda at all. For example, in a variant of the *trains* domain where fruit has to be processed and transported, intermediate locations for cars, engines, and fruit can be derived.

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 be inconsistent, i.e., unsolvable. For example, *holding(a)* and *holding(c)* cannot be simultaneously achieved 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 15 shows the performance of IPP generating the optimal plan using a goal agenda refined with intermediate goals for roundtrip problems from the briefcase domain where an increasing number of objects from 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 15: IPP on briefcase roundtrip problems using a goal agenda with intermediate goals (+G+I), RIFO (+R), and linearization (+L).

7 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 of our current research activities. Furthermore, the combination of various definitions of $\prec$ and their potential to speed up planners should be worth exploring. Besides this, we are exploring ways to embed goal agendas into the search process, i.e., to on-line order goals.

Appendix

The woodshop domain contains the 8 operators sand, vacuum, paint, forklift, drill, saw, put-in-box, unpack. Drilling damages surfaces and can only be performed on objects with non-round shapes, painting requires that objects should be free of dust. A typical planning task is the following:

```
object:    obj1;
color:     green plain;
shape:     blob square round;
surface:   rough smooth;
dimension: three-eighths;

initial:
shape(obj1 round) painted(obj1 plain) surface(obj1 rough);


goal:
shape(obj1 round) surface(obj1 smooth) painted(obj1 green)
has-hole(obj1  three-eighths) !dusty(obj1) boxed(obj1);
```

Running the analysis procedure yields the following goal agenda, which reflects the natural ordering over subgoals that exists:

```
step  1: has-hole(obj1,three-eighths)
step  2: surface(obj1,smooth)  shape(obj1,round)
step  3: not-dusty(obj1) painted(obj1,green)
step  4: boxed(obj1)
```

for which the optimal plan of 9 actions is found when incrementally planning for the increasing goal sets from the agenda

```
time step   0: saw(obj1,blob,round)
time step   1: drill(obj1,three-eighths)
time step   2: saw(obj1,round,blob)
time step   3: sand(obj1,smooth)
time step   4: vacuum(obj1)
time step   5: forklift(obj1)
time step   6: paint(green)
time step   7: forklift(obj1)
time step   8: put-in-box(obj1)
```

The scheduling world contains more complex actions, but is similar in the sense that a natural ordering exists in which objects have to be processed. For the following planning problem

```
object:     A B;
paint:      red blue;
location:   sched store;
paint-cond: nil red blue;
shape-cond: oblong cylindrical flat rectangular;
temp-cond:  cold hot;
surf-cond:  rough smooth polished;
width:      null null-1 null-2 null-3;
```

```
orient:     nil front back left-side right-side;

initial:
shape(A rectangular) temperature(A cold)
surface-condition(A rough) painted(A blue)
has-hole(A null-2 front) has-hole(A null-3 left-side)
shape(B flat) temperature(B hot) surface-condition(B rough)
painted(B nil) has-hole(B null nil) have-supply(red)  is-at(sched);

goal:
shape(A oblong) painted(A red) has-hole(A null-1 right-side)
not has-hole(A null-2 front) not has-hole(A null-3 left-side)
shape(B cylindrical) temperature(B cold) surface-condition(B polished)
painted(B red) has-hole(B null-2 back);
```

an agenda with 4 entries is generated:

```
step  1: shape(B,cylindrical)
step  2: not-has-hole(A,null-3,left)
         not-has-hole(A,null-2,front)
step  3: painted(B,red)
step  4: shape(A,oblong)
         has-hole(A,null-1,right)
         painted(A,red)
         surface-condition(B,polished)
         temperature(B,cold)
         has-hole(B,null-2,back)
```

Although this agenda does not completely order goals to the maximal possible extent, because too many $\prec$ relations cause some subgoal sets to collapse, it reduces runtime significantly and still enables the planner to generate the optimal plan of 17 actions. In general it holds that ordering subgoals maximally increases the danger of superfluous actions in plans, while non-maximal agendas can still significantly improve runtime.

```
time step   0: roll(B)
time step   1: roll(A)
time step   2: make-sprayable(red)
               cool-down(B)
time step   3: spray-paint(B,red)
time step   4: move(sched,store)
time step   5: buy-paint(red)
time step   6: move(store,sched)
time step   7: make-sprayable(red)
               punch(B,null-2,back)
               cool-down(A)
time step   8: cool-down(B)
               lathe(A)
time step   9: polish(B)
               cool-down(A)
time step  10: spray-paint(A,red)
time step  11: punch(A,null-1,right)
```

References

[BF95] A. Blum and M. Furst. Fast planning through planning graph analysis. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1636–1642. Morgan Kaufmann, San Francisco, CA, 1995.

[BW93] A. Barrett and D. Weld. Characterizing subgoal interactions for planning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1388–1393. Morgan Kaufmann, San Francisco, CA, 1993.

[Cha87] D. Chapman. Planning for conjunctive goals. *AIJ*, 32(3):333–377, 1987.

[CI89] J. Cheng and K. Irani. Ordering problem subgoals. In IJCAI-89 [IJC89], pages 931–936.

[DC89] M. Drummond and K. Currie. Goal ordering in partially ordered plans. In IJCAI-89 [IJC89], pages 960–965.

[DNK97] Y. Dimopoulos, B. Nebel, and J. Koehler. Encoding planning problems in non-monotonic logic programs. In Steel [Ste97], pages 169–181.

[EKR96] A. El-Kholy and Barry Richards. Temporal and resource reasoning in planning: the parcPLAN approch. In W. Wahlster, editor, *Proceedings of the 12th European Conference on Artificial Intelligence*, pages 614–618. John Wiley & Sons, Chichester, New York, 1996.

[GS96] A. Gerevini and L. Schubert. Accelerating partial-order planners: Some techniques for effective search control and pruning. *Journal of Artificial Intelligence Research*, 5:95–137, 1996.

[HH89] J. Hertzberg and A. Horz. Towards a theory of conflict detection and resolution in nonlinear plans. In IJCAI-89 [IJC89], pages 937–942.

[IC87] K. Irani and J. Cheng. Subgoal ordering and goal augmentation for heuristic problem solving. In D. McDermott, editor, *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 1018–1024, Milan, Italy, August 1987. Morgan Kaufmann.

[IJC89] *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989. Morgan Kaufmann.

[JR90] D. Joslin and J. Roach. A theoretical analysis of conjunctive-goal problems. *Artificial Intelligence*, 41:97–106, 1990.

[KNHD97] J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos. Extending planning graphs to an ADL subset. In Steel [Ste97], pages 273–285.

[Koe96] J. Koehler. Planning from second principles. *Artificial Intelligence*, 87(1-2):148–187, 1996.

[Kor87] R. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88, 1987.

[KS96] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the 14th National Conference of the American Association for Artificial Intelligence*, pages 1194–1201. AAAI Press, 1996.

[NDK97] B. Nebel, Y. Dimopoulos, and J. Koehler. Ignoring irrelevant facts and operators in plan generation. In Steel [Ste97], pages 338–350.

[PJP97] M. Pollack, D. Joslin, and M. Paolucci. Selection strategies for partial-order planning. *Journal of Artificial Intelligence Research*, 6:223–262, 1997.

[Ste97] S. Steel, editor. *Proceedings of the 4th European Conference on Planning*, volume 1348 of *LNAI*. Springer, 1997.