Characterizing Subgoal Interactions for Planning

Anthony Barrett and Daniel S. Weld
Department of Computer Science and Engineering*
University of Washington, Seattle, WA 98195
{barrett, weld}@cs.washington.edu

Abstract

Korf's taxonomy of subgoal interactions [5] fails to differentiate between classes that have vastly different computational properties; in particular, some sets of serializable goals are easy to solve while others are difficult. We present a predictive theory of planner performance based on the number of feasible serializable orderings. The central notions are the classes of trivially serializable and laboriously serializable subgoals which are tractable and difficult respectively.

To illustrate our theory, we compare the interaction structure of three planners. We demonstrate a domain whose structure is trivially serializable for a partial order planner, laboriously serializable for a total order planner, and nonserializable for a world-state, regression planner. As predicted, only the partial order algorithm could plan for this domain. We conclude that our theory is partially confirmed.

1 Introduction

Despite considerable research over past decades, many topics in planning remain a black art. In particular, it is often difficult to predict the performance of a given planner on a new domain without lengthy empirical tests. Recently, Korf has suggested that subgoal interactions are a good predictor for planning performance [5]. However, while Korf's seminal analysis of independent, serializable, and nonserializable collections of subgoals [5] forms the foundation for a theory of planning performance, Korf's interaction taxonomy fails to differentiate between classes that have vastly different computational properties. In particular, we have found [1] that some classes of serializable subgoals can be quickly solved while others are intractable. In this paper we introduce a predictive theory of planning performance by refining Korf's taxonomy with the following classes:

- A set of subgoals is trivially serializable if each subgoal can be solved sequentially in any order without ever violating past progress. We show that trivial serializability is more general than independence, yet results in comparable performance.
- A set of subgoals is laboriously serializable if there exists a 1 order in which the subgoals can not be solved without possibly violating past progress. When it is difficult to determine the correct order, on average laboriously serializable subgoals are as intractable as nonserializable ones.

In extensive empirical studies reported elsewhere [1], we found complete correlation between planning performance in a domain and the type of subgoal interactions in that domain. Each planner performed well only when dealing with problems whose subgoals were independent or trivially serializable; problems with laboriously serializable or nonserializable subgoals were intractable.

In this paper, we argue that our extended hierarchy of subgoal interactions (figure 2) allows a crisp analysis of the performance of multiple planning algorithms on a variety of domains. To buttress our claims, we analyze three different commitment strategies for ordering decisions in planning. We consider three different planning algorithms: POCL, a partial order algorithm devised by McAllester and Rosenblitt [6], TOCL a similar algorithm which commits to a total order, and TOPI, a plan-state algorithm which is isomorphic to a regression search through the space of world-states.

We construct an artificial domain (D₅,S⁺) to differentiate between the planning algorithms. D₅,S⁺'s subgoals are trivially serializable for POCL, laboriously serializable for TOCL, and nonserializable for TOPI. Our theory predicts that POCL should perform well on problems in this domain but that the total order planners should branch intractably. We ran empirical tests on randomly generated problems which confirmed these predictions. Thus we conclude that there is supporting evidence for our claims of a predictive theory.

In the next section we formally define the class of planning problems that we consider and describe the three planning algorithms. In section 3, we extend Korf's taxonomy of subgoal interactions and cast it in the context of search through the space of incomplete plans. Then, in section 4 we demonstrate that the representations used by the different planners engender different classes of subgoal interactions for a given problem domain. Section 5 confirms our predictions of the planners' empirical behavior. Finally, section 6 discusses related work and 7 summarizes our results.

*This research is funded in part by National Science Foundation Grant IRI-8657362, Office of Naval Research Grant 99-J-1994, and a grant from the Xerox corporation.

1388 Planning
2 Planning Algorithms

Like [5], we view planning as a search problem. In order to discuss our planners, we need to define a planning problem and how it can be considered as a search process. From [6] we adopt:

Definition 1 A STRIPS OPERATOR consists of an operator name plus a precondition list, an add list and a delete list. The elements of the precondition, add, and delete lists are all function-free, atomic expressions. A STRIPS PLANNING PROBLEM is a triple <O, Σ, D> in which O denotes a set of STRIPS operators, Σ denotes a set of initial propositions, and D denotes a set of goal propositions.

2.1 Plan-State Searches

Previous analyses of planning problems [3; 5] viewed planning as a search through a graph of world-states — i.e. a graph in which nodes are labeled with a set of propositions that specify what is true in that state of the world. If an operator’s preconditions are satisfied in a world-state, then an edge leads from that node to the node denoting the effect of applying that operator. One of the major contributions of Sacerdoti’s NOAH [11] was a conceptual shift: instead of viewing planning as search through a space of world-states, NOAH searched through a space of (possibly incomplete) plan-states. Our planners perform similar searches.

Definition 2 A PLAN-STATE is a triple: <S, O, B> in which S denotes a set of plan steps (also known as actions), O denotes a set of ordering constraints that specify a (possibly partial) order on S, and B denotes a set of binding constraints over the variables mentioned by the steps in S.

Reformulation in terms of plan-states makes the structure of a search space dependent on the planner as well as the domain. The arcs between world-states represent feasible domain actions, but the arcs between plan-states represent feasible extensions to an incomplete plan (i.e., the addition of a step, ordering constraint or binding constraint).

In plan-state search, a planning problem is encoded in an initial plan-state <S, O, B> consisting of two steps s₀ and s₁. The step s₀ adds Σ and ε₀ has O for preconditions. This plan-state has no variable-binding constraints, but O has one constraint to force s₀ before s₁. The set of operators, O, defines the possible steps which can be added to the set of steps, S, in a plan-state. The goal of the search is to find a solution plan-state.

Definition 3 A SOLUTION PLAN-STATE <S, O, B> is a plan-state in which the preconditions of each step sᵢ ∈ S are all necessarily true in the input situation of sᵢ.

To make this definition precise, we recall the following terminology from Chapman’s formalization of planning [2]. The input situation of a step is a set of propositions that are true immediately prior to the execution of that step. A proposition is necessarily true in the input situation of a step in a partially ordered plan-state <S, O, B> when it is true in all completions that extend the partial constraints of O and B into total constraints.

2.2 Algorithms

In order to test our intuitions regarding how representations of the space of plan-states affects planning difficulty, we implemented three different planning algorithms. Each of these algorithms is sound, complete, and exhibits what McAllester terms the “systematic” property [6]. Loosely speaking, systematicity means that totally ordered lists of steps is required to plan states in an irreduntant fashion — visiting every possible plan-state exactly once. This property is reflected in the structure of each planner’s search space in that the directed edges form a tree rooted in the initial plan-state.

The first algorithm, called POCL, is a lifted version of McAllester’s algorithm. It uses a partially ordered step representation for defining plans, and is shown in figure 1. POCL is the plan (i.e., in record the steps). Introducing a step into a plan and to protect that purpose. If a step Sᵢ adds a proposition p to satisfy a precondition of step Sⱼ, then Sᵢ·Sⱼ denotes the causal link. In a plan-state, a link Sᵢ·Sⱼ is threatened if some existing step Sᵢ can be ordered between Sᵢ and Sⱼ, and Sⱼ either deletes or adds a proposition that can be unified with p by adding variable-binding constraints without making the plan-state’s variable constraint set inconsistent.

In an invocation of POCL, each precondition p of a plan-state Sⱼ is an open condition if it has no corresponding causal link Sᵢ·Sⱼ ∈ L. POCL searches for a solution plan-state by eliminating open conditions in G while ensuring the safety of causal links in L.

The second algorithm, TOPL, is similar to POCL, but it restricts its use of plan-states by only generating plans with totally ordered lists of steps. In each among plan-states, NOAH searched through a space of (possibly incomplete) plan-states. Our planners perform similar searches.

TOPL is a simple regression planner like the one in [9]. It only adds steps to the beginning of the plan (i.e., in record the steps). Although TOPL does not require data structures for links or step ordering, it shares with the other algorithms the data structures and routines for variable bindings and constraints.

3 Characterizing Subgoal Interactions

Like [3] and [5] we measure the difficulty of a problem in terms of how hard it is to break it up into subproblems and use these subproblems to guide the search for a final solution.

1We note that the utility of systematicity has not been clearly documented. We suspect that for some domains systematic planners will be more efficient than nonsystematic algorithms while the converse will hold in other domains. Since a detailed evaluation of the utility of systematicity is beyond the scope of this paper (but see [4]), we feel that holding systematicity constant in our experiments increases their validity.

2A COMMON Lisp implementation of this algorithm, known as SNIP, has become quite popular as a framework for AI research and education. Send mail to bug-analysts, washngton, to acquire the source code for the three planners and for test domains.

Barrett and Weld 1389
Algorithm: POCL(\(\preceq S, O, B_L, G_L\))
1. Termination: If \(G = 0\), report success and stop.
2. Goal selection: Let \(e\) be a proposition in \(G\), and let \(S_{\text{next}}\) be the step for which \(e\) is a precondition.
3. Operator selection: Let \(S_{\text{old}}\) be an existing step, or some new step, that adds \(e\) before \(S_{\text{next}}\). If no such step exists or can be added then backtrack.
   Let \(L' = L \cup \{S_{\text{old}} \rightarrow S_{\text{next}}\}\), \(S' = S \cup \{S_{\text{old}}\}\), \(O' = O \cup \{S_{\text{old}} < S_{\text{new}}\}\) and \(B' = B \cup \{S_{\text{old}} < S_{\text{new}}\}\). This set of variable bindings makes \(S_{\text{next}}\) add \(e\). Backtrack point: Each existing and possibly addable step must be considered for completeness.
4. Update goal set: Let \(G' = (G - \{e\}) \cup \) preconditions of \(S_{\text{old}}\), if new.
5. Causal link protection: A step \(s_2\) threatens a causal link \(s_1 \rightarrow s_2\), when it can occur between \(s_1\) and \(s_2\) and add or delete \(p\). For every step \(s_2\) that might threaten a causal link \(s_1 \rightarrow s_2\), \(s_2 \in L'\):
   - Ensure that \(s_2\) does not threaten \(s_1 \rightarrow s_1\) by adding constraints to \(O'\) and/or \(B'\). Backtrack point: Each way to protect \(s_1 \rightarrow s_2\) from \(s_2\) must be considered for completeness.

Figure 1: The Partial-Order, Causal-Link Algorithm

3.2 Korf’s Subgoal Hierarchy

In the simplest case a subgoal is an intermediate state on the path from initial state to goal. Intuitively, it is clear that searching from the initial state to the subgoal and then again from the subgoal to the goal might be faster than searching all the way in one step. Korf presents a broader definition which we adopt for this paper.

...we formally define a subgoal to be a set of states, with the interpretation that a state is an element of a subgoal set if and only if it has properties that satisfy the subgoal [5, page 68].

It is frequently awkward to refer to subgoals explicitly as sets of states. A common technique used in [3] and [5] is to use goal propositions (i.e., elements of \(O\)) to specify subgoals. A world-state is in a subgoal if the subgoal’s associated goal proposition is true in the world-state.

The conceptual shift to planning with plan-states affects the way subgoals are specified. When searching the space of plan-states we define subgoals in terms of the achievability of various elements of \(O\).

Definition 4 Let \(\preceq O, \Sigma, \Omega\) be a STRIPS planning problem and \(P \in \Omega\) denote one of the goal propositions. The subgoal specified by \(P\) with respect to \(\preceq O, \Sigma, \Omega\) is a set, \(U\), of plan-states such that every \(\preceq O, \Sigma, \Omega\) satisfies:
1. \(S\) contains a step \(s_0\) which only adds \(\Sigma\).
2. \(S\) contains a step \(s_n\) which only requires \(\Omega\).
3. Every total order of \(S\) consistent with \(O\) has \(s_0\) and \(s_n\) as the first and last steps, respectively.
4. \(P\) is necessarily true in the input situation of \(s_0\).

Proving that a proposition \(P\) is necessarily true can be done using Chapman’s modal truth criterion [2]. Certain properties of our algorithms make this proof process easier. For example, in TOPI the operator selection step
Since nonserializable subgoals may need to be violated and reestablished many times, they offer little guidance to a planner: solution time will likely rise superlinearly with the number of subgoals.

3.3 Extending the Interaction Hierarchy

Extensive empirical experiments [1] showed that Korf's hierarchy was not detailed enough to predict the performance of planning algorithms in a domain. In particular, some sets of serializable subgoals are easy to solve while others are intractable. Furthermore, the notion of complete subgoal independence is so restrictive that it seldom occurs in practice. We address this weakness by refining the interaction taxonomy:

Definition 5 A set of subgoals is trivially serializable if they can be solved in any order without ever violating a previously solved subgoal.

We note the following important properties:

Proposition 1 Let \( U \) and \( V \) be sets of subgoals such that \( V \subseteq U \).

1. If \( U \) is serializable then \( V \) may be serializable or nonserializable.

2. If \( U \) is trivially serializable then \( V \) is trivially serializable.

Proof: An observation of Korf's [5] regarding the Sussman Anomaly proves part 1. He showed that the goal set \( \{ (\text{on A B}), (\text{on B C}) \} \) is nonserializable (in the space of world-states), but the superset \( \{ (\text{on A B}), (\text{on B C}), (\text{on C table}) \} \) is serializable. The proof of part 2 is straightforward.

Proposition 1 illustrates why arbitrary serializable goals can be difficult — when subsets are considered (e.g., in localized decision making) then mistakes can be made which require backtracking over previously satisfied subgoals. This isn't true when subgoals are trivially serializable. Furthermore, trivial serializability is more general than independence because it is strictly a topological property while independence is metric (i.e., defined in terms of a distance function). In particular, two trivially serializable subgoals are not independent if some operator helps to achieve both of them. Of course there is a price for the extra generality — trivial serializability does not carry the compositional properties that are entailed by independence. In particular, solutions to the separate subgoals cannot necessarily be concatenated to achieve a solution to the conjunct. Nevertheless, trivial serializability is much more common than independence and it appears to make the complexity of planning linear in the number of subgoals.

Just as trivially serialized subgoals represent a real collection, it is natural to consider collections of subgoals that are pathological while still being serializable. For such a collection of subgoals, it is difficult to determine the correct order. Solving problems with laboriously serializable subgoals, without prior knowledge of the order, may take time exponential in the number of subgoals.

Definition 6 A set of \( n \) subgoals is laboriously serializable if there exists at least one serializable ordering yet at least \( \frac{1}{n} \) of the subgoal orders can not be solved sequentially without possibly violating a previously solved subgoal.

Even if the majority of subgoal orderings are fine, the exponential cost of backtracking on the few pathological cases will dominate average planning time. Since bad orderings can require exponentially more time than good orderings, tractability requires that the number of bad orderings be exponentially decreasing in the number of orderings. But if \( \leq \) (or any only polynomially decreasing percentage) of the orderings are bad, then intractability will dominate. We summarize our extended taxonomy in figure 2.

Figure 2: An Extended Hierarchy of Subgoal Collections.

4 Analyzing Planning Performance

In this section we present a problem domain which induces a different class of subgoal interactions for each of the three planners. Subgoals in this domain are nonserializable for TOP, laboriously serializable for TOCL, yet trivially serializable for POCL. As with other domains that we have tested [1], this classification was predictive of empirical results: only the partial order planner was able to quickly solve randomly generated problems in this domain. These problems consist of \( n + 1 \) possible goal conditions, \( G_1 \) and \( G_2 \) through \( G_n \), which are achieved with \( n + 2 \) possible STRIPS actions:

\[
\begin{align*}
\text{(def-step :action A1 :precond \{L\} :add \{G_1\} :delete \{L_{\text{not}}\})} \\
\text{(def-step :action A2 :precond \{P\} :add \{G_1\} :delete \{P\})} \\
\text{(def-step :action A3 :precond \{L\} :add \{P\} :delete \{\})}
\end{align*}
\]

We call this domain \( D^S \) because it was created from \( D^S \) (reported in [1]) by adding the actions and conditions that are subscripted with an asterisk. The steps interact with each other through their delete lists in a manner which ensures that only a single ordering of steps will achieve a given problem. As illustrated in figure 3 steps \( A_1 \) or \( A_2 \) would threaten the precondition link of the step immediately above it, and all steps \( A_i \) threaten the precondition of step \( A_{i+1} \).

Proposition 2 The subgoals specified by the problem conjuncts (i.e., specified by \( G_1 \) and \( G_2 \)) in the \( D^S \) domain are nonserializable for TOP.

Proof: Consider the subgoals for conditions \( G_1 \) and \( G_2 \). Achieving \( G_1 \) requires adding step \( A_1 \), and achieving \( G_2 \) requires adding steps \( A_2^i \) and \( A_3^j \). Since \( A_2^i \)
Figure 3: The causal structure of a solution to an n + 1
goal problem

deletes A1, it must be added before A3. Since A2
is irrelevant to other subgoals, G3 must be ordered
before G4. Similarly, step A1's deleting I, leads to
the requirement that G1 must be ordered before G2.
Since these two constraints are contradictory, there
are no valid subgoal orderings.

The causal link algorithms differ from TOPI in that
they focus on a single subgoal and, upon reaching that
subgoal, move on to the next subgoal in the ordering.

Proposition 3 The subgoals specified by the problem
conjuncts in the D^2S^1s domain are trivially serializable
for POCL.

Proof: Consider the plan-state reached by focusing on
the first m subgoals. Suppose that its m steps are
ordered such that A1 precedes A_{m+1}. This is trivially
true of the initial plan-state. Achieving the m + 1
subgoal involves adding a step, and protecting
causal links ensures that all m + 1 steps in the new
plan-state are ordered such that A1 precedes A_{m+1}.
Since this partial ordering is consistent, POCL can
always achieve the next subgoal without violating a
previous subgoal.

Similarly, achieving the subgoal specified by G4 can
be done at any point in this arbitrary ordering. The
step A_2 is just like any step A_1, and protecting the
link for I, always forces A^2 before any step A_1.

Proposition 4 The subgoals specified by the problem
conjuncts in the D^2S^1s domain are laboriously serializable
for TOCL.

Proof: Consider orderings of the subgoals specified by
G4 through G_5, and let the first subgoal be specified
by G4. The next subgoal in the ordering must be
specified by G_6, or by G_{14} + 1. When TOCL at-
ttempts to satisfy the subgoal specified by G7 for
j - i ≠ 1, the steps A_1 and A_j will appear uncon-
strained with respect to each other because they do
not affect each others preconditions. This lack of
an ordering constraint gives the linearization step
license to choose an arbitrary order. However, since
the steps in the final solution are totally ordered,
only one of these arbitrary orders is the correct one.
This means that subgoal G4 may have to be vi-
olated, via backtracking, to find the correct order.

In general the valid serialization orders are those
where a subgoal G1 only appears at the beginning
or after the appearance of G_{1,2} or G_{14}. Any other
ordering would have a case like the one described
above. There are \( \binom{2}{i-1} \) such orders that start

Figure 4: As predicted POCL outperformed both total
order planners in D^2S^1s.

with G4, and the total number of such orders is
\( 2^n - 1 \).

Consider including the subgoal specified by G4. The
subgoals are still serializable. Any serialization or-
dering of the subgoals specified by G4 through G_5
becomes a serialization ordering for the entire set of
subgoals when the subgoal for G_5 is inserted after
G4's subgoal, or at the beginning of the ordering
that starts with G4. This makes the total number of
serialization orderings \( 2^n \). Since \( 2^n \) is a small
fraction of \( (n+1)! \) for large \( n \), the subgoals are la-
boriously serializable.

5 Empirical Results

Our theory predicts that both total order planners
should find problems in D^2S^1s intractable, while the
partial order planner should have no difficulty. We
tested our predictions by generating 390 problems at 13
different difficulty levels (30 problems per problem type).
Each problem type contained 13 initial conditions and
had a difficulty of between 13 and 15 goal conditions.
The initial and goal conditions were randomly permuted for
each problem, and performance was measured in seconds of
Sun SparcStation CPU time. The results (mean CPU time
and 95% confidence intervals) appear in figure
4. As predicted, only POCL existed.

Interestingly, [6] describes a problem which is similar
to D^2S^1s, but in which subgoals are trivially serializ-
able for both TOCL and POCL. In this domain, TOCL
performed slightly better than the partial order planner
which increases the confidence in our theory.

This paper discusses an artificial (some would say
"toy") domain, but our results are quite general. Al-
though the subgoal structure of more realistic domains
are messier than D^2S^1s, preliminary studies [1] show
that the notions of trivial and laborious serializability
will facilitate the engineering of practical planning ap-
lications by pinpointing the subgoal interactions that
are likely to be costly for whatever planning algorithm
is applied.

6 Related Work

As mentioned previously, our work builds on Korf's [5]
taxonomy of subgoals and domain complexity. Jeslin
subgoals* with a topological analysis of subgoals in terms of their connected components. Our extensions to Korf's taxonomy are independent of Joslin and Roach's contribution since they do not consider the number of viable subgoal ordering classes. This is the key concept underlying our notions of trivial and laborious serializability.

Our investigation of partial order planners is matched by Minton et al. [7] who analyze the TO and UA algorithms which resemble the trivially SERIALIZABLE and LABORIOUSLY SERIALIZABLE of our TOCL and POCL algorithms with one difference: unlike our planners, TO and UA are not systematic. In a recent extension to their earlier work, Minton et al. [8] consider the effects of different search strategies for distribution of solutions on performance. We have also done extensive empirical studies; our results are detailed in [1].

7 Conclusions

We have presented a predictive theory of planning performance and supported the theory with experiments on three algorithms which employed different strategies for delaying step ordering decisions. Our paper makes four contributions:

- We demonstrate that Korf's [5] subgoal taxonomy fails to differentiate between classes that have vastly different computational properties. Both TOCL and POCL have serializable subgoals in $D^*S^*$ yet one branch is intrinsically while the other displays apparently linear performance.

- We present a new, predictive theory of planning performance that emphasizes the number of feasible serialization orderings. The cornerstone of our theory are the numbers of trivially SERIALIZABLE and LABORIOUSLY SERIALIZABLE sets of subgoals. Since all orderings of trivially serializable subgoals lead to a global solution, this class is computationally tractable. We note that pure trivial and pure laborious serializability are but two points on a continuum of arduousness and many real problems may be intermediate in difficulty.

- We demonstrate our theory by analyzing commitment strategies for ordering decisions. We considered three planning algorithms and presented a new domain, $D^*S^*$ which distinguished between them. Since the natural subgoal decomposition of a $D^*S^*$ problem is trivially serializable for POCL, laboriously serializable for TOCL and nonserializable for TOPI, we predicted that only the partial order planner would have acceptable performance in this domain. An empirical study confirmed this prediction providing a measure of confidence in the theory. We note that our theory is consistent with the numerous additional studies presented in [1].

- Our theory provides a new way to view the advantage of the partial order plan representation. All planners perform well only when confronted with trivially serializable subgoals, but more domains are laboriously serializable for partial order planners than for the total order planners.

Much remains to be done. Our results promise a complete search strategy; extending the analysis to aggressive algorithms would be interesting. Our artificial domain was easy to analyze because all problems in that domain had the same subgoal interaction class for a given planner. In real world domains, however, a planner is likely to find some problems trivially serializable and others laboriously serializable. Extending our theory to heterogeneous subgoal interactions would be interesting.

Also, we have only tested our theory on algorithms handling the STRIPS representation. We plan to use UCPOP [10] to explore whether partial-order representations are useful given more expressive domains which include conditional effects and universally quantified effects. We suspect that our theory will predict UCPOP to be more efficient than totally ordered variants, but this intuition awaits validation.

ACKNOWLEDGEMENTS

This research was greatly improved by discussions with and comments by Paul Cohen, Ernest Davis, Oren Etzioni, Steve Hanks, James Henderson, Raa Kambhampoti, Craig Knoblock, Neal Laird, David McAllester, Steve Minton, Edwin Pednault, Ying Sun, John Tenenberg, Brian Williams, and Mike Williamson. We thank Steven Soderland, with whom we started this project.

References


Barrett and Weld 1393