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 serialization 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 sets 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 $\frac{1}{n}$ orders 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 (D^1S^1*) to differentiate between the planning algorithms. D^1S^1* 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-8957302, Office of Naval Research Grant 90-J-1904, and a grant from the Xerox corporation.

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* $\prec \mathcal{O}, \Sigma, \Omega \succ$ *in which* $\mathcal{O}$ *denotes a set of STRIPS operators,* Σ *denotes a set of initial propositions, and* Ω *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:* $\prec S, O, B \succ$ *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* $\prec S, O, B \succ$ consisting of two steps s_0 and s_∞. The step s_0 adds Σ, and s_∞ has Ω for preconditions. This plan-state has no variable-binding constraints, but O has one constraint to force s_0 before s_∞. The set of operators, $\mathcal{O}$, 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 $\prec S, O, B \succ$ *is a plan-state in which the preconditions of each step* $s_i \in S$ *are all necessarily true in the input situation of* s_i.

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 $\prec S, O, B \succ$ 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][1]. Loosely speaking, systematicity means that each algorithm is guaranteed to search among plan-states in an irredundant 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.[2] It uses a partially ordered step representation for defining plans, and is shown in figure 1. *POCL* uses *causal links* to record the purpose for introducing a step into a plan and to protect that purpose. If a step S_i adds a proposition p to satisfy a precondition of step S_j, then $S_i \xrightarrow{p} S_j$ denotes the causal link. In a plan-state, a link $S_i \xrightarrow{p} S_j$ is *threatened* if some existing step S_k can be ordered between S_i and S_j, and S_k either deletes or adds[3] 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 step S_j is an *open condition* if it has no corresponding causal link $S_i \xrightarrow{p} S_j \in L$. *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, *TOCL*, is similar to *POCL*, but it restricts its use of plan-states by only generating plans with totally ordered sets of steps. It can insert steps anywhere between s_0 and s_∞ in a plan, but it can never reorder two existing steps. The algorithm is created from *POCL* by adding a step to linearize the plan immediately prior to the recursive call, and optimizing the other steps to take advantage of the total-order plan.

The third algorithm, *TOPI*, is a simple regression planner like the one in [9]. It only adds steps to the beginning of the plan (i.e., immediately after s_0). Although *TOPI* 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.

[1] We 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.

[2] A COMMON LISP implementation of this algorithm, known as *SNLP*, has become quite popular as a framework for AI research and education. Send mail to `bug-snlp@cs.washington.edu` to acquire the source code for the three planners and for test domains.

[3] Steps that add p threaten the causal link $S_i \xrightarrow{p} S_j$ because they negate the *purpose* for adding step S_i. *POCL* would not be systematic if it ignored these threats.

Algorithm: $POCL(\prec S, O, B \succ, G, L)$

1. **Termination:** If $G = \emptyset$, report success and stop.
2. **Goal selection:** Let c be a proposition in G, and let S_{need} be the step for which c is a precondition.
3. **Operator selection:** Let S_{add} be an existing step, or some new step, that adds c before S_{need}. If no such step exists or can be added then backtrack. Let $L' = L \cup \{S_{add} \xrightarrow{c} S_{need}\}$, $S' = S \cup \{S_{add}\}$, $O' = O \cup \{S_{add} \prec S_{need}\}$, and $B' = B \cup$ the set of variable bindings to make S_{add} add c. *Backtrack point: Each existing and possibly addable step must be considered for completeness.*
4. **Update goal set:** Let $G' = (G - \{c\}) \cup$ preconditions of S_{add}, if new.
5. **Causal link protection:** A step s_k *threatens* a causal link $s_i \xrightarrow{p} s_j$ when it can occur between s_i and s_j and add or delete p. For every step s_k that might threaten a causal link $s_i \xrightarrow{p} s_j \in L'$:
   - Ensure that s_k does not threaten $s_i \xrightarrow{p} s_j$ by adding constraints to O' and/or B'. *Backtrack point: Each way to protect $s_i \xrightarrow{p} s_j$ from s_k must be considered for completeness.*
6. **Recursive call:** $POCL(\prec S', O', B' \succ, G', L')$.

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

3.1 Subgoals

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 Ω) 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 Ω.

Definition 4 *Let $\prec \mathcal{O}, \Sigma, \Omega \succ$ be a STRIPS planning problem and let $P \in \Omega$ denote one of the goal propositions. The* SUBGOAL SPECIFIED BY P WITH RESPECT TO $\prec \mathcal{O}, \Sigma, \Omega \succ$ *is a set, U, of plan-states such that every $\prec S, O, B \succ \in U$ satisfies:*

1. *S contains a step s_0 which only adds Σ.*
2. *S contains a step s_∞ which only requires Ω.*
3. *Every total order of S consistent with O has s_0 and s_∞ as the first and last steps, respectively.*
4. *P is necessarily true in the input situation of s_∞.*

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 has the restriction that added steps cannot delete open goals. This and the fact that *TOPI* can only add steps to the beginning of an incomplete plan ensures that a proposition is necessarily true when it is either in the initial conditions or it is added by a step and all of that step's preconditions are necessarily true. Thus a goal proposition P is necessarily true once all of the open preconditions of steps added to solve P are in the initial conditions.

The causal-link protection-step of *TOCL* and *POCL* makes it easy to prove the necessary truth of a proposition. A goal proposition P is necessarily true if it has an associated causal link $s_i \xrightarrow{P} s_\infty$, and all of the preconditions step s_i are necessarily true. The causal-link-protection step ensures that no steps ever interfere with the truth of P. Thus P is necessarily true once all of preconditions of all of the steps involved in solving P have associated causal links.

3.2 Korf's Subgoal Hierarchy

Frequently, a problem is difficult enough to make it necessary to specify several subgoals in the effort to guide search. Korf classifies a set of subgoals in terms of how the members interact with each other.

Independent subgoals are the rare ideal case — progress toward one has no effect on another. We follow Korf's definition:

> A collection of subgoals are INDEPENDENT if each operator only changes the distance to a single subgoal. ... One of the important properties of independent subgoals, which is clear from the definition, is that an optimal global solution can be achieved by simply concatenating together optimal solutions to the individual subproblems in any order [5, page 71].

Solving a single independent subgoal might be nontrivial, but the complexity of problems with independent subgoals increases only linearly with the number of subgoals. Korf defines serializable subgoals, those that do interact in a limited manner, with the following statements.

> We define a set of subgoals to be SERIALIZABLE if there exists an ordering among the subgoals such that the subgoals can always be solved sequentially without ever violating a previously solved subgoal in the order [5, page 71].

Thus, serializability means that for every state in the intersection of the first n subgoals there exists a path to a state in the $n+1$st subgoal that lies wholly within the intersection. Since these paths are ways to reach later subgoals without interfering with those previously achieved, each subgoal only has to be established once. Of course, using the wrong order can lead to having to violate and reestablish a subgoal an exponential number of times.

Korf labels all sets of subgoals that aren't serializable as nonserializable:

> It is often the case that given a collection of subgoals, previously satisfied subgoals must be violated in order to make further progress towards the main goal, regardless of the solution order. Such a collection of subgoals will be called NON-SERIALIZABLE [5, pages 72–73].

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 $\mathcal{U}$ and $\mathcal{V}$ be sets of subgoals such that $\mathcal{V} \subseteq \mathcal{U}$.*

1. *If $\mathcal{U}$ is serializable then $\mathcal{V}$ may be serializable or nonserializable.*
2. *If $\mathcal{U}$ is trivially serializable then $\mathcal{V}$ is trivially serializable.*

Proof: An observation of Korf's [5] regarding the Sussman Anomaly proves part 1. He showed that the goal set {(on A B), (on B C)} is nonserializable (in the space of world-states), but the superset {(on A B), (on B C), (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 an ideal 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 $\frac{1}{n}$ (or any only polynomially decreasing percentage) of the orderings are bad, then intractability will dominate. We summarize our extended taxonomy in figure 2.

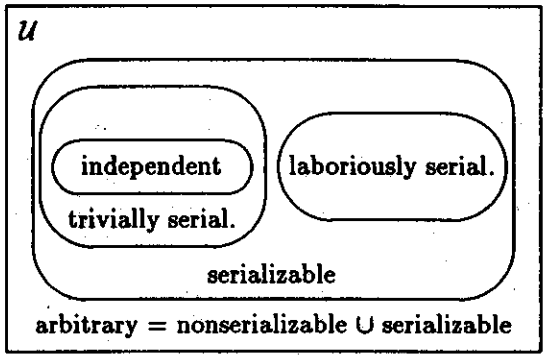

Figure 2: An Extended Hierarchy of Subgoal Collections.

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 *TOPI*, 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_* and G_1 through G_n, which are achieved with $n+2$ possible STRIPS actions:

```
(def-step  :action A_i  :precond {I_i}  :add {G_i}
           :delete {I_{i+1}, I_*})
(def-step  :action A_*^2  :precond {P_*}  :add {G_*}
           :delete {I_1})
(def-step  :action A_*^1  :precond {I_*}  :add {P_*}
           :delete {})
```

We call this domain D^1S^1* because it was created from D^1S^1 (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_i, 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_*^1.

Proposition 2 *The subgoals specified by the problem conjuncts (i.e., specified by G_j and by G_*) in the D^1S^1* domain are nonserializable for TOPI.*

Proof: Consider the subgoals for conditions G_1 and G_*. Achieving G_1 requires adding step A_1, and achieving G_* requires adding steps A_*^1 and A_*^2. Since A_*^2

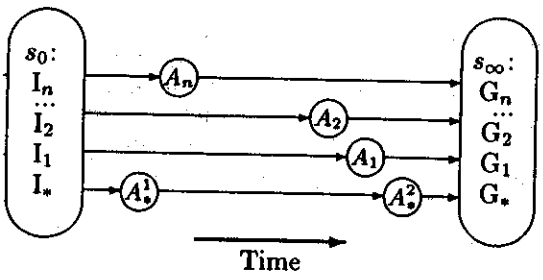

Figure 3: The causal structure of a solution to an $n+1$ goal problem

deletes I_1, it must be added before A_1. Since A_*^2 is irrelevant to other subgoals, G_* must be ordered before G_1. Similarly, step A_1's deleting I_* leads to the requirement that G_1 must be ordered before G_*. 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^1S^1* 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 A_i precedes A_{i+1}. This is trivially true of the *initial plan-state*. Achieving the $m+1$st subgoal involves adding a step, and protecting causal links ensures that all $m+1$ steps in the new plan-state are ordered such that A_i precedes A_{i+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 G_* can be done at any point in this arbitrary ordering. The step A_*^2 is just like any step A_i, and protecting the link for I_* always forces A_*^1 before any step A_i. □

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

Proof: Consider orderings of the subgoals specified by G_1 through G_n, and let the first subgoal be specified by G_i. The next subgoal in the ordering must be specified by G_{i-1} or by G_{i+1}. When *TOCL* attempts to satisfy the subgoal specified by G_j for $|j-i| \neq 1$, the steps A_i and A_j will appear unconstrained 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 G_j may have to be violated, via backtracking, to find the correct order.

In general the valid serialization orders are those where a subgoal G_i only appears at the beginning or after the appearance of G_{i-1} or G_{i+1}. Any other ordering would have a case like the one described above. There are $\binom{n-1}{k-1}$ such orders that start with G_k, and the total number of such orders is 2^{n-1}.

Consider including the subgoal specified by G_*. The subgoals are still serializable. Any serialization ordering of the subgoals specified by G_1 through G_n becomes a serialization ordering for the entire set of subgoals when the subgoal for G_* is inserted after G_1's subgoal, or at the beginning of the ordering that starts with G_1. 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 laboriously serializable. □

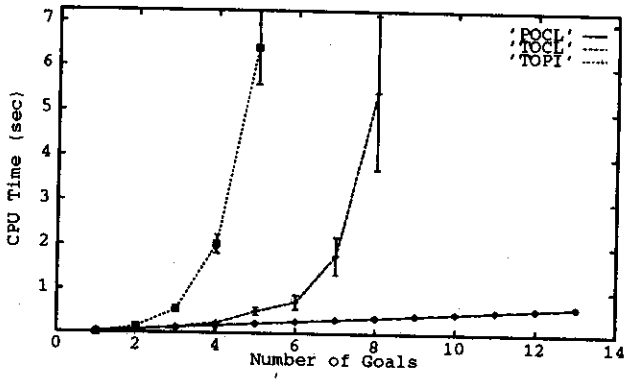

Figure 4: As predicted *POCL* outperformed both total order planners in D^1S^1*.

5 Empirical Results

Our theory predicts that both total order planners should find problems in D^1S^1* 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 1 and 13 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 90% confidence intervals) appear in figure 4. As predicted, only *POCL* excelled.

Interestingly, [1] describes a domain which is similar to D^1S^1*, but in which subgoals are trivially serializable 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. Although the subgoal structure of more realistic domains are messier than D^1S^1*, preliminary studies [1] show that the notions of trivial and laborious serializability will facilitate the engineering of practical planning applications 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. Joslin and Roach [3] extend Korf's analysis of nonserializable

subgoals[4] 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 orderings while 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 propositional versions 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 and the 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^1S^1* yet one branches intractably 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 centerpiece of our theory are the classes 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^1S^1* which distinguished between them. Since the natural subgoal decomposition of a D^1S^1* problem is trivially serializable for *POCL*, laboriously serializable for *TOCL* and nonserializable for *TOPI*, we predicted that only the partial ordered 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 trivially serializable for partial order planners than for the total order planners.

Much remains to be done. Our results presume 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 restrictive 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.

[4] Unfortunately, Joslin and Roach do not phrase their work in these terms, appearing unaware of Korf's work.

ACKNOWLEDGEMENTS

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

References

[1] A. Barrett and D. Weld. Partial Order Planning: Evaluating Possible Efficiency Gains. *Aritificial Intelligence*, To appear in 1993.

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

[3] D. Joslin and J. Roach. A Theoretical Analysis of Conjunctive-Goal Problems. *Artificial Intelligence*, 41:97–106, 1989/90.

[4] S. Kambhampati. Characterizing Multi Contributor Causal Structures for Planning. In *Proceedings of the First International Conference on AI Planning Systems*, pages 116–125, June 1992.

[5] R. Korf. Planning as Search: A Quantitative Approach. *Artificial Intelligence*, 33(1), September 1987.

[6] D. McAllester and D. Rosenblitt. Systematic Nonlinear Planning. In *Proceedings of AAAI-91*, pages 634–639, July 1991.

[7] S. Minton, J. Bresina, and M. Drummond. Commitment Strategies in Planning: A Comparative Analysis. In *Proceedings of IJCAI-91*, pages 259–265, August 1991.

[8] S. Minton, M. Drummond, J. Bresina, and A. Phillips. Total Order vs. Partial Order Planning: Factors Influencing Performance. In *Proceedings of KR-92*, October 1992.

[9] N. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Company, Palo Alto, CA, 1980.

[10] J.S. Penberthy and D. Weld. UCPOP: A Sound, Complete, Partial Order Planner for ADL. In *Proceedings of KR-92*, pages 103–114, October 1992.

[11] E. Sacerdoti. The Nonlinear Nature of Plans. In *Proceedings of IJCAI-75*, pages 206–214, 1975.