DOCUMENT RESUME

ED 353 954 IR 015 913

AUTHOR Gratch, Jonathan; DeJong, Gerald
TITLE An Analysis of Learning To Plan as a Search Problem.
INSTITUTION Illinois Univ., Urbana. Dept. of Computer Science.
SPONS AGENCY National Science Foundation, Washington, D.C.
REPORT NO UILU-ENG-92-1703; UIUCDCS-R-92-1723
PUB DATE Jan 92
CONTRACT NSF-IRI-87-19766
NOTE 19p.; For related reports, see IR 015 914-915.
PUB TYPE Information Analyses (070) -- Reports - Research/Technical (143)

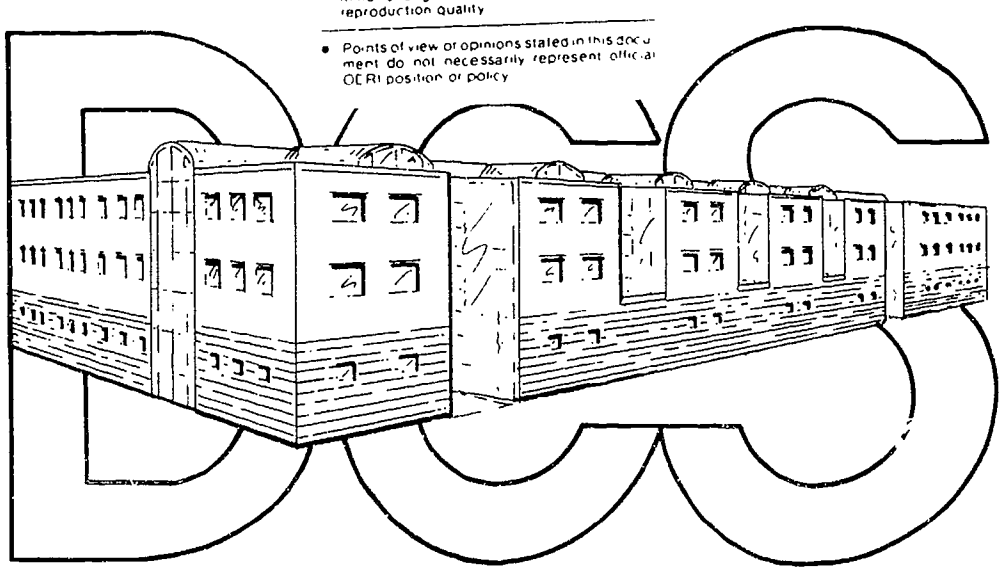
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Artificial Intelligence; *Computer System Design; Evaluation Methods; *Learning Strategies; *Planning; Problem Solving; *Search Strategies; Systems Development
IDENTIFIERS *Empirical Research; Machine Learning

ABSTRACT

Increasingly, machine learning is entertained as a mechanism for improving the efficiency of planning systems. Research in this area has generated an impressive battery of techniques and a growing body of empirical successes. Unfortunately the formal properties of these systems are not well understood. This is highlighted by a growing corpus of demonstrations where learning actually degrades planning performance. In this paper we view learning to plan as a search problem. Learning is seen as a transformational process where a planner is tailored to a particular domain and problem distribution. To accomplish this task, learning systems draw from a vocabulary of transformation operators such as macro-operators or control rules. These "learning operators" define a space of possible transformations through which a system must search for an efficient planner. This study shows that the complexity of this search precludes a general solution and can only be approached via simplifications. (Frequently unarticulated commitments which underlie current learning approaches are illustrated.) These simplifications improve learning efficiency but not without tradeoffs. In some cases these tradeoffs result in less than optimal behavior. In others, they produce planners which become worse through learning. It is hoped that by articulating these commitments we can better understand their ramifications can be better understood. Finally, a particular learning technique--COMPOSER--is discussed which explicitly utilizes these simplifications to ensure performance improvements with reasonable efficiency. (Contains 34 references.) (Author/ALF)

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN



THE NEW ADDITION

REPORT NO. UIUCDCS-R-92-1723 UILU-ENG-92-1703

An Analysis of Learning to Plan as a Search Problem

by

Jonathan Gratch
Gerald DeJong

January 1992

An Analysis of Learning to Plan as a Search Problem

Jonathan Gratch and Gerald DeJong

Artificial Intelligence Research Group
Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana–Champaign
405 North Matthews Avenue
Urbana, IL 61801

Telephone: (217) 244–1503
Internet: gratch@cs.uiuc.edu

Keywords

Learning and Planning
Empirical methods
Theoritical and Empirical evaluation

Abstract

Increasingly, machine learning is entertained as a mechanism for improving the efficiency of planning systems. Research in this area has generated an impressive battery of techniques and a growing body of empirical successes. Unfortunately the formal properties of these systems are not well understood. This is highlighted by a growing corpus of demonstrations where learning actually *degrades* planning performance. In this paper we view learning to plan as a search problem. Learning is seen as a transformational process where a planner is tailored to a particular domain and problem distribution. To accomplish this task, learning systems draw from a vocabulary of transformation operators such as macro–operators or control rules. These "learning operators" define a space of possible transformations through which a system must search for a efficient planner. We show that the complexity of this search precludes a general solution and can only be approached via simplifications. We illustrate the frequently unarticulated commitments which underly current learning approaches. These simplifications improve learning efficiency but not without tradeoffs. In some cases these tradeoffs result in less than optimal behavior. In others, they produce planners which become worse through learning. It is hoped that by articulating these commitments we can better understand their ramifications. Finally, we discusses a particular learning technique which explicitly utilizes these simplifications to ensures performance improvements with reasonable efficiency.

This research is supported by the National Science Foundation, grant NSF IRI 87–19766

1 INTRODUCTION

In machine learning there is considerable interest in techniques which improve planning ability. Investigation in this area has identified a wide array of techniques including macro–operators [DeJong86, Fikes72, Mitchell86], chunks [Laird86], and control rules [Minton88, Mitchell83]. With these techniques comes a growing battery of successful demonstrations in domains ranging from 8–puzzle to Space Shuttle payload processing. Unfortunately, the formal properties of these techniques are not well understood. This is highlighted by a growing body of demonstrations where learning degrades planning performance [Etzioni90a, Gratch91b, Minton85, Subramanian90].

In this paper we will develop a view of learning as search and compare alternate learning strategies against this common perspective. We will show that the complexity of general task of improving a planner must be approached by imposing simplifications. Section 3 explores the frequently unarticulated commitments which underly learning approaches. These simplifications improve learning efficiency but not without tradeoffs. In some cases these tradeoffs result in less than optimal behavior. In others, they lead to planners which grow worse through learning. It is hoped that by articulating these commitments we can better understand their ramifications.

Section 4 discusses a particular method which ensures performance improvements with reasonable efficiency. The COMPOSER system is an extension of the PRODIGY/EBL approach to speed–up learning. Its contribution is a rigorous form of utility analysis which provides probabilistic guarantees of improvement.

2 ADEQUATE LEARNING

A learning system should take a particular planning system operating within a particular domain and tailor it to more effectively solve problems. This can be viewed as a transformational process where a series of transformations are applied to the original problem solver (see [Gratch90b, Greiner92]). A planner may be transformed by the addition of control knowledge. Different forms of control knowledge include macro–operators [Braverman88, Laird86, Markovitch89], control rules [Drummond89, Etzioni90a, Minton88, Mitchell83], and static board evaluation functions [Utgoff91]. Alternatively, a learning system may modify the domain theory. Such transformations could be truth (accuracy) preserving (as in conjunct reordering or deletion of redundancy [Smith85], or non–truth preserving (as in theory revision tasks [Richards91, Towell90]).

The transformations available to a learner define its *vocabulary of transformations*. These are essentially learning operators and collectively they define a *transformation space*. For instance, acquiring a macro–operator can be viewed as transforming the initial state (the original planner) into a new state (the planner operating with the macro–operator). A learning system must search this space for a sequence of transformation which result in a better planner.

First, we will precisely define what makes a learning system *adequate*. Given an initial planner, P_0, a learning system is *minimally adequate* if it: 1) halts without making any changes, or 2) applies a sequence of transformations which yields a new planner P_{new} which is preferred to P_{old} (i.e. if it does anything, it produces a better planner). One learning system is *more adequate* (better) than another if it produces preferred planners. A system which identifies the most preferred planner in the transformation space (there may be more than one) is *optimal*. To make this notion concrete we must define a preference function over planners.

There are many ways to assign preferences. For this paper we view preference in terms of a numeric *utility function* which ranks differently transformed planners according to our intuitive notions of preferences. In particular, we will consider the case where a utility value can be assigned to a planner's behavior for any given problem. The utility of a planner is then defined with respect to a particular problem distribution as the sum of problem utilities weighted by the probability of each problem. A similar definition appears in [Greiner92]. This covers the obvious measures of effectiveness. For example, if we are interested in minimizing problem solving cost, problem utility increases as the cost required to solve that problem decreases. The utility of the planner is then:

$$UTILITY(planner_i) = -\sum_{prob \in Distribution} Cost(planner_i, prob) \times Pr(prob)$$

Utility is a preference function which ranks different planners. But the learning algorithm must search through alternative transformations. We need a way to convert preferences over planners into preferences over transformations. The *utility of a transformation* is the change in utility that results from applying the transformation to a particular planner. This means the utility of a transformation is conditional on the planner to which it is applied. We denote this as: ΔUTILITY(Transformation|Planner). Applying a transformation with positive utility results in a more effective planner. A learning system need not explicitly compute utility values to identify preferred planners, but it must act (at

least approximately) as if it does. In fact, most learning systems do not explicitly make utility determinations.

3 CHALLENGES AND DIRECTIONS

We are presenting a view of learning as search. A learning system must move through the transformation space in search of preferred planners. To accomplish this task it must identify transformations with positive utility. Typically, transformations are proposed in response to problem solving success or impasses, and their benefit is estimated (if at all) from future problem solving episodes.

There are three basic challenges to adequate learning: 1) the space of possible transformations is too large, 2) it is expensive to reliably identify good transformations, and 3) it may be infeasible to solve even a single problem with the initial planner — confounding the processes of transformation proposal and validation. These difficulties suggest there there does not exist a general solution to this task. In spite of this, there are a number of published techniques which claim to work well. The remainder of this section resolves this contradiction by illustrating how learning techniques make simplifications to address each challenge.

In very few cases are simplifications explicit in the published reports of these works. Thus, it is difficult to determine the precise simplifications an algorithm embodies. Even if elucidated, these may not useful in the sense that it would be hard to instantiate them in a different algorithm. Instead, we present concise and obvious simplifications to each challenge. We then argue how different techniques are best viewed as approximating these commitments. Figure 1 summarizes the approaches to be discussed. This is intended to be representative of the approaches which are popular in the literature and is not meant to be exhaustive. The discussion is organized into approaches for each of the three basic challenges. Unless otherwise noted, solutions to one challenge are independent of the solutions to other challenges. There are frequently relationships between the approaches within each challenge, and these will be noted in the text.

3.1 Transformation Space Complexity

In this section we presume that the utility of transformations can be determined. The challenge is to efficiently find an appropriate sequence of transformation. Unfortunately, there are many ways to change the initial planner. We can consider all possible subsets of transformations. Furthermore,

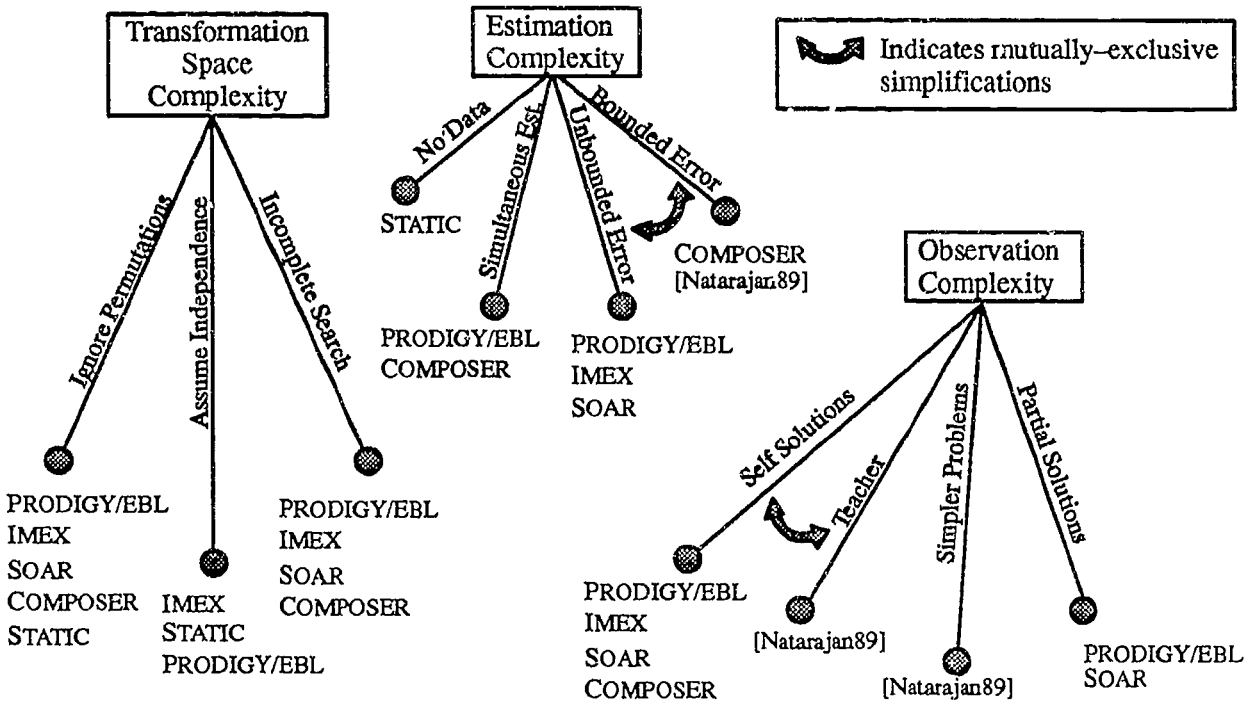


Figure 1: Summary of simplifying approaches

as several authors have noted, the order of these transformations is often important (e.g. very different behavior can result from changes in the order in which macro operators are evaluated).

For a set of n transformations there are $O(nn!)$ distinguishable changes (any planner may be the result of up to n ordered transformations). Even n may be excessively large. For example, Etzioni provides a bound on the number of control rules entertained by his STATIC system which is exponential in the number of predicates in the domain theory [Etzioni90b pp. 171–174]. If transformations involve continuous parameters (e.g. modifications to the real–valued parameters of a static board evaluator), the space may be uncountably infinite.

An unconstrained algorithm will consider all $nn!$ alternatives. This is rarely feasible. However, there are a number of simplifications which can reduce the space. We discuss three which are orthogonal and many system adopt more than one. The first approach is to avoid searching all possible orderings for transformations.

AVOIDING PERMUTATIONS (T1)

With this commitment, the number of distinguishable planners is reduced from $O(nn!)$ to $O(2^n)$. In practice this means that if an algorithm is considering a new transformation to adopt, it will not con-

sider all possible places to insert the transformation into the existing sequence. In some cases the order is resolved in a particular way. In the case of macro–operators, Shavlik suggests one ordering scheme: order the library of macro–operators such that each newly learned macro–operators is placed before the original domain theory but after previously learned macro–operators [Shavlik88]. Minton's PRODIGY/EBL system arbitrarily maintains control rules in the order they were learned [Minton88 p. 79]. In both cases the authors acknowledge that different ordering schemes result in different planning behavior.

Legislating a particular ordering will avoid many alternatives. When we can demonstrate that the order of transformations is irrelevant, or that the best ordering can be determined without search, this simplification can be adopted without cost. Without such guarantees, the system may not find preferred planners when they exist. Thus it may preclude optimality or otherwise lower the adequacy of the learning system. On the other hand, a system utilizing this simplification retains minimally adequacy, as it does not effect the reliability of utility values. Some systems which ignore alternative permutations are IMEX [Braverman88], COMPOSER [Gratch91a], STATIC [Etzioni90a], BAGGER [Shavlik88], and PRODIGY/EBL [Minton88]. We are not aware of a technique which does not adopt this assumption.

While T1 reduces complexity, it is generally insufficient. One of the difficulties is that even with ordering constraints there are a large number of conditional utilities: $\Delta UTILITY(T_1|P)$, $\Delta UTILITY(T_1|T_2P)$, $\Delta UTILITY(T_1|T_2T_3P)$, etc. A powerful simplification is to treat the utility of a transformation as independent of other transformations.

$$\textit{INDEPENDENCE:}\quad \forall T_i|T_j,\ \Delta UTILITY(T_i|T_jP) = \Delta UTILITY(T_i|P) \qquad \text{(T2)}$$

Under this simplification a learning system can evaluate each transformation once, without regard to context of other transformations. This allows a local decision procedure for adopting transformations. Each transformation is evaluated without regard for other adopted transformations. If the transformation is beneficial, adopt it, otherwise discard it. The complexity of search is reduced to $O(n)$.

This assumption leads to adequate learning under certain sufficient conditions. If the transformations are in fact independent, a learning system can discover the optimal transformation sequence in $O(n)$. If this condition cannot be guaranteed, weaker conditions are sufficient to display at least

minimal adequacy. A learning system can ignore dependencies as long as negative interactions are not overwhelming. An example of an overwhelming negative interaction would be the case where, individually, two transformations benefit the planner but, collectively, they hurt performance.

Many speed–up learning systems adopt transformations using decision procedures which do not explicitly consider interactions (including STATIC [Etzioni90a], PRODIGY/EBL, RECEBG [Letovsky90], IMEX, and PEBL [Eskey90]). Ma and Wilkins illustrate a similar situation for knowledge–base revision systems [Wilkins89]. Systems which do not adopt this assumption include PALO [Greiner92], COMPOSER, and [Leckie91].

Unfortunately, accounts of systems which ignore interactions do not characterize when this simplification is appropriate. Furthermore, it is becoming clear that the transformations utilized by these techniques do interact. In [Gratch90b] and [Gratch91b] we demonstrate how macro operators and control rules produce harmful interactions. In the later article we illustrate a simple domain which exhibits strong negative interactions. When tested on this domain, the utility analysis technique of PRODIGY/EBL and the nonrecursive hypothesis of STATIC generate planners several times *worse* than the initial planner. These results show this simplification can produce inadequate performance. They also highlight why more emphasis must be placed on explicating and providing sufficient conditions for the simplifications which underlie learning techniques.

A final simplification is to avoid exploring the entire search space:

INCOMPLETE/UNRECOVERABLE SEARCH (T3)

There exists a wide array of search techniques to deal with complexity. For example, an unrecoverable technique like hillclimbing can reduce the complexity to $O(n^2)$ (at most n choices at each step for at most n steps). The number of choices can be further reduced by only considering a subset of the n possible transformations. In either case these simplifications retain minimal adequacy at the cost of reduced learning opportunities. A hill–climbing technique which is stuck at a local maxima may produce a planner which is less adequate than a more comprehensive learner.

Many learning techniques consider only a subset of the legal transformations. For example, SOAR does not entertain all possible chunks, but only those learned in response to impasses. Other techniques are explicit hill–climbers which terminate search when a local maxima is reached [Greiner92] or when the training set is exhausted [Gratch91a].

3.2 Estimation Complexity

The preceding discussion assumed that utility values were available to the learning system. Unfortunately this is rarely the case. The utility of a transformation depends on information which is frequently unavailable like the distribution of future problems. The natural approach is to estimate utility from training examples.

The simplest approach to estimation is estimation by brute force. Execute the planner over all problems in the expected test set, observing its behavior. Then make a transformation and rerun the planner on the same set of problems. Use the difference between the two runs to estimate the utility of the transformation. As the test set may be large, unavailable, or infinite, and there may be many transformations from which to choose from, this approach is impractical. Many techniques are thriftier in their use of examples. They embody the following simplification:

SIMULTANEOUS ESTIMATION (E1)

This means that the technique extracts information about multiple transformations from each example. PRODIGY/EBL, COMPOSER, SYLLOG [Markovitch89], and PALO [Greiner92] perform simultaneous estimation. This simplification is justified if the estimate for one transformation is not influenced by whatever other transformations are being estimated. This condition is easily verified empirically, and in some systems the simplification is unjustified. For example, PRODIGY/EBL gathers statistics for control rules as other rules are learned and forgotten. These shifting conditions influence the estimates. In [Gratch91b] we illustrate a domain where these influences lead to learning behavior which is not minimally adequate. COMPOSER and PALO were designed to explicitly enable simultaneous estimation.

The primary difficulty, however, is that we typically cannot feasibly evaluate a transformation over all future problems. Instead we must estimate utility from a subset of the distribution. Approaches to this problem vary in the number of examples required to produce their estimates. By definition, estimates are approximations to reality. A transformation which is estimated to be good may in fact have negative utility. Many approaches to not attempt to quantify or bound the this type of error.

UNQUANTIFIED/UNBOUNDED ERROR (E2)

Techniques which derive utility estimates from examples but do not reason about the accuracy of those estimates including PRODIGY/EBL, SYLLOG, and [Leckie91]. The decision of how many

examples are necessary is left in the hands of the user and no guarantees are provided for the accuracy of the resulting estimates. It may be difficult in these cases to provide sufficient conditions for when the probability of error is acceptably low. The complement of E2 is:

QUANTIFIED/BOUNDED ERROR (E3)

Statistical methods can bound the probability of mistakes. Error can be reduced by taking more examples. In fact, given a pre–specified level of error, there are techniques which can draw sufficient examples to reach this error level (assuming the method is appropriate). The guarantee is gained at the cost of many training examples, but Greiner and Cohen show how (under weak assumptions) to achieve an arbitrary level of confidence with polynomial examples [Greiner92]. Different statistical models require differing amounts of data. There is a large body of work in statistics which describes these alternative methods and their sufficient conditions.

Finally, the most demanding simplification is:

LEARNING WITHOUT EXAMPLES (E4)

If we can identify transformations which reduce the problem solving cost of all possible problems, we can learn without data. This is similar to the notion of *dominance* in [Wellman90]. If this property can be demonstrated, a transformation will yield an improved planner independent of the problem distribution. Such transformations do exist. For example, a planner can only benefit from deleting an unsatisfiable or redundant operator. The STATIC system is grounded in the principle that its set of transformations will increase utility independent of distribution [Etzioni90b p. 134]. Unfortunately, the STATIC approach does not quantify or bound its accuracy and it is difficult to demonstrate conditions when the accuracy is acceptable. Etzioni illustrates some domains where STATIC is adequate but negative results appear in [Gratch91b]. But when it is appropriate, learning without example is a powerful simplification as it precludes the challenge of observation complexity to which we now turn.

3.3 Observation Complexity

Learning techniques must observe data from the task environment to estimate utility values. Besides the distribution of problems, this data often involves properties of the planner's state space, or properties of solution paths. The most common way to gather this data is to solve problems with the cur-

rently transformed planner. Problems can be randomly drawn from the distribution and solution their traces examined.

LEARNING FROM SELF–SOLUTIONS (O1)

But planning is not tractable in general. Thus, it may not be feasible to solve even one problem with the initial planner. Learning from self–solutions equates observation complexity with the complexity of the initial planner in the domain of interest. If this approach is to be feasible, this complexity must be sufficiently small. While this condition is rarely articulated, it is implicit in a wide range of learning techniques [Braverman88, Laird86, Minton85, Mitchell83, Ruby91].

One might ask why we need to learn at all if problem solving is already feasible. However, in many circumstances this quite reasonable. For instance, in situations where large numbers of problems must be solved, small increases in efficiency can result in huge savings.

If learning from self–solutions is impractical, the alternative approach is to:

LEARNING FROM A TEACHER (O2)

Some techniques require a teacher to provide the appropriate data. This places the observation complexity in the hands of the teacher. The teacher, though its superior knowledge, presumably can elicit the data in reasonable time. For example of this approach, Tadepalli demonstrates that when domains are *serially decomposable*, a system can generate a polynomial–time problem solver by learning a body of control knowledge called a macro–table [Tadepalli91]. To learn this table the system requires a teacher which knows the macro–table we wish to learn. This teacher then provides the learning system with solutions generated according to the table. Natarajan introduces a technique which requires the teacher to provide optimal solution paths [Natarajan89].

LEARNING FROM SIMPLER PROBLEMS (O3)

An intriguing alternative is to learn to solve intractable problems by training on simpler, tractable problems. This is analogous to classroom learning were a carefully selected set of "text book" problems should lead to sophisticated problem solving ability. Natarajan demonstrates some sufficient conditions under which this type of learning is possible [Natarajan89].

When it is not feasible to completely solve a problem, useful information may still be gleaned from partial solutions:

LEARNING FROM PARTIAL SOLUTIONS (O4)

Techniques which learn from failure like SOAR and PRODIGY/EBL can generate transformations in response to incomplete solution attempts. Unfortunately it is difficult to generate reliable utility estimates from partial solution traces. For example, it may be the case that a transformation behaves very differently as problem solving proceeds. By prematurely terminating the planner, we may not get a representative sample of the transformations behavior.

4 COMPOSER

We have identified a number of the challenges facing an adequate learner. In this section we describe in some detail a particular approach to this problem, the COMPOSER system. An important property of this technique is that it makes its simplifications explicit. It implements a preference criteria based on reducing planning time. The utility of transformations is explicitly estimated and the system adopts transformations which, with high probability, have positive utility. The approach is probabilistically adequate, where the probability of error is arbitrary and determined by the user.

COMPOSER is currently implemented with control rules as its vocabulary of transformations. A number of simplifications are adopted to approach the problem. The technique incrementally builds strategies one transformation at a time. The system only considers adding new control rules to the end of the existing list. Thus, alternative orders for each transformation are ignored (T1). Only a subset of legal transformations are considered, namely rules which are proposed in response to planning impasses (T3). This reduced space of transformations is explored by hill–climbing (T3). Multiple transformations are estimated simultaneously (E1). Utility estimates are based on a statistical model and error is bounded to a user specified value (E3). The technique gathers data through self–solution (O1). If problem solving is not feasible, system terminates without adopting any transformations.

COMPOSER can be viewed as an rigorous version of the utility analysis approach of PRODIGY/EBL. While our implementation is an augmentation of PRODIGY/EBL, the basic approach can be extended to a wide range of planners and transformation vocabularies.

4.1 Gathering Observations

COMPOSER uses the mechanisms of PRODIGY/EBL to propose control rules for the PRODIGY planner. The PRODIGY planner is a STRIPS–like planner which generates an annotated record of problem solving. The problem trace describes the resources spent at each node, including time spent evaluating control rules. The PRODIGY/EBL's learning module analyzes this trace and conjectures possible control rules.

In PRODIGY/EBL, conjectured rules are added to the planner and undergo a heuristic form of utility analysis. Rules which fail this analysis are retracted. COMPOSER replaces this heuristic analysis with a rigorous alternative. Conjectured transformations are placed on a *pending list* of rules. The preconditions of pending rules are evaluated and the match cost recorded. However, the recommendations of pending rules are not performed. Instead, the system annotates the problem trace with the changes they would have introduced. In this way, estimated rules do not influence the behavior of the planner which enables data to be gathered on multiple rules from a single solution trace. The specific process is fairly involved and is described in [Gratch90a]. The extracted utility values are conditional on the current set of transformations adopted by the system.

4.2 Estimating Utility

Utility values from several problems can be combined to estimate the utility of a transformation across the distribution of problems. The system should only adopt transformations which have positive utility. Additionally, the system should remove from the pending list any transformations with negative utility. In statistics this is referred to as a *sequential analysis problem.* Observations are gathered until some criteria, a *stopping criteria*, is satisfied. In this case we are estimating the utility of transformations to some specified confidence. We require the user to provided an error parameter, δ, which specifies the acceptable probability of incorrectly adopting a transformation.[1]

COMPOSER must choose among two hypotheses for any rule on the pending list:

$$H_0: \Delta\text{UTILITY(rule|planner)} <= 0, \text{ or } H_1: \Delta\text{UTILITY(rule|planner)} > 0$$

Averaging the observations across problems yields an estimate of the true utility. This estimate will differ from the true value, so the system must bound the discrepancy. In particular, if the rule is negative, the system must bound the probability that it will appear positive and vice versa. This is equiva-

1. Alternatively we could require that $1 - \delta$ represent the confidence that every step makes progress. This requires determining a δ_i at each step such that the sum of all δ_i's equals δ.

lent to bounding the probability that the magnitude of the true utility is exceeded by the difference between true and estimated utility:

$$P(\,|\text{ESTIMATE} - \text{UTILITY}| > |\text{UTILITY}|\,) = \delta$$

Nádas [Nadas69] describes a distribution–free stopping criteria which can be applied.[2] It solves the more general problem of bounding the probability that an unknown mean is in an interval of size proportional to the mean. In our case we bound the unknown mean to an open interval which is exactly the size of the unknown mean. The technique requires gathering M examples where M is defined as:

$$M = \min_{n>1} \{n: (V_n/\overline{X})^2 \leq n(p/a)^2\}$$

where $V_n = \sum (X_i - \overline{X})^2$, $\overline{X}$ is the average utility, X_i is the utility on problem i, and p is the proportional parameter. In our case p is close to one (0.9999). The parameter a satisfies the constraint that $\Phi(a) = \delta/2$, where Φ is the cumulative distribution function of the standard normal distribution.

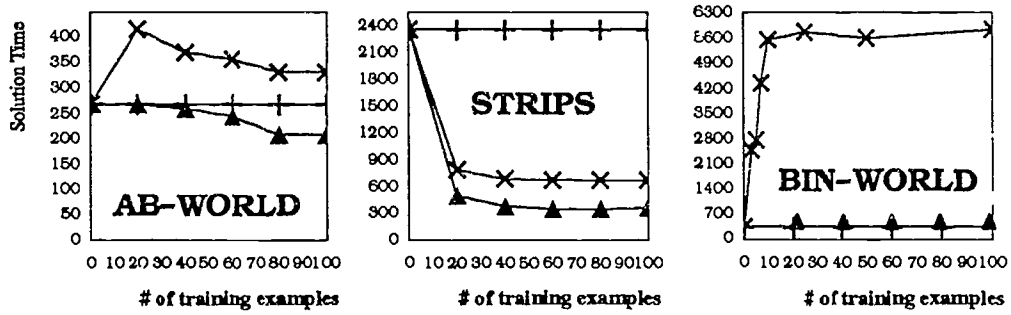
4.3 Learning

The stopping criteria permits COMPOSER to identify beneficial transformations with high probability. After each problem solving attempt, COMPOSER updates the statistics and evaluates the stopping criteria for each element of the pending list. If no control rule has attained the confidence requirement, another problem is solved. If the stopping rule identifies control rules with positive utility (there may be more than one), COMPOSER adds the control rule with highest positive utility to the current strategy, and removes it from the pending list. Statistics for the remaining pending rules are discarded as they are meaningless in the context of the resulting control strategy. If the stopping criteria identifies pending rules with negative utility, they are eliminated from the pending list. Eliminating a pending rule does not affect the current strategy, so the statistics associated with the remaining pending rules are left unchanged. This cycle is repeated until the training set is exhausted. Each time a transformation is adopted the efficiency of the PRODIGY planner is increased, giving COMPOSER an *anytime* behavior [Dean88].

2. The method is limited to distributions with a finite variance and provides an approximate confidence interval. Woodroofe provides second–order results that this approximation is very close in practice [Woodroofe82]. Greiner and Cohen [Greiner92] provide an alternative stopping rule which provides somewhat stronger guarantees at the cost of many more training examples. For example, in the domains we have tested, Nádas' technique requires on the order of ten training examples to accurately estimate the utility of a transformation. In the same domains, Greiner and Cohen's method requires several thousand training examples per transformation.

4.4 Evaluating COMPOSER

COMPOSER was tested on a domain from [Minton88], a domain in [Etzioni90a] for which PRODIGY/EBL produced harmful strategies, and a domain in [Gratch91b] which yielded detrimental results for both STATIC and PRODIGY/EBL. The results are summarized in Figure 2. The confidence for adding a transformation was set at 95%. In each domain the system is trained on a 100 training examples drawn according to a fixed distribution. Snapshots are taken at designated intervals to provide a learning curve (the complete procedure is described in [Gratch90a]). The graphs illustrate learning curves where the independent measure is the number of random training examples and the dependent measure is execution time for 100 test problems. More effective strategies have *lower* solution times. We provide PRODIGY without learning and PRODIGY/EBL as benchmarks for comparison.



DOMAIN	▲ COMPOSER			✕ PRODIGY/EBL			✚ No Learning
	Rules Learned (average)	Learning Time (average)	Solution Time (average)	Rules Learned (average)	Learning Time (average)	Solution Time (average)	Solution Time (average)
AB-WORLD	1	1663 sec.	208 sec.	11	1252 sec.	331 sec.	268 sec.
STRIPS	4	4139 sec.	357 sec.	20	3773 sec.	673 sec.	2362 sec.
BIN-WORLD	0	3425 sec.	346 sec.	2	6383 sec.	6020 sec.	346 sec.

Figure 2: Summary of empriical results

The results illustrates several interesting features. On all domains COMPOSER exceeded the performance of PRODIGY/EBL, including domains where PRODIGY/EBL is inadequate. This is significant as both systems investigate the same space of transformations. An important result is that the intermediate planning times resulting from COMPOSER are monotonically decreasing. This indicates that conditional utility is accurately estimated. A surprising fact is that in the domains where COMPOSER acquired a strategy, only one or two control rules account for most, if not all, of the savings. This suggests that most of the rules acquired by PRODIGY/EBL are, at best, superfluous.

6 CONCLUSIONS

Learning shows great promise to extend the generality and effectiveness of planning techniques. But if learning is to be useful, we must explicitly characterize the properties of these systems. This article introduces the notion of adequacy to assess the merits of learning techniques. Surprisingly, many learning systems are not even minimally adequate in that they may worsen planning performance.

The complexity of learning makes unconstrained techniques infeasible. But the task can be approached by introducing one or more simplifications. We discussed how many learning techniques can be viewed as implicitly adopting these simplifications. In many cases these constraints are only approximately satisfied with the result that these systems are not even minimally adequate. This highlights the need for explicit examination of the assumptions underlying new learning systems, as well as the need for analytical as well as empirical justification in future research. We discussed the COMPOSER system as one approach which is explicitly justified.

References

[Braverman88] M. S. Braverman and S. J. Russell, "IMEX: Overcoming intractability in explanation based learning," *Proceedings of the National Conference on Artificial Intelligence*, St. Paul, MN, 1988, pp. 575–579.

[DeJong86] G. F. DeJong and R. J. Mooney, "Explanation–Based Learning: An Alternative View," *Machine Learning 1*, 2 (April 1986), pp. 145–176. (Also appears as Technical Report UILU–ENG–86–2208, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana–Champaign.)

[Dean88] T. Dean and M. Boddy, "An Analysis of Time–Dependent Planning," *Proceedings of The Seventh National Conference on Artificial Intelligence*, Saint Paul, MN, August 1988, pp. 49–54.

[Drummond89] M. Drummond, "Situated Control Rules," *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, Ontario, Canada, May 1989, pp. 103–113.

[Eskey90] M. Eskey and M. Zweben, "Learning Search COntrol for Constraint–Based Scheduling," *Proceedings of the National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 908–915.

[Etzioni90a] O. Etzioni, "Why Prodigy/EBL Works," *Proceedings of the National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 916–922.

[Etzioni90b] O. Etzioni, "A Structural Theory of Search Control," Ph.D. Thesis, Department of Computer Science, Carnegie–Mellon University, Pittsburgh, PA, In preparation, 1990.

[Fikes72] R. E. Fikes, P. E. Hart and N. J. Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence 3*, 4 (1972), pp. 251–288.

[Gratch90a] J. Gratch and G. DeJong, "Utility Generalization and Composability Problems in Explanation–Based Learning," Technical Report UIUUCDCS–R–91–1681, Department of Computer Science, University of Illinois at Urbana–Champaign, 1990.

[Gratch90b] J. Gratch and G. DeJong, "A Framework for Evaluating Search Control Strategies," *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, San Diego, CA, November 1990, pp. 337–347.

[Gratch91a] J. Gratch and G. DeJong, "A Hybrid Approach to Guaranteed Effective Control Strategies," *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, IL, June 1991.

[Gratch91b] J. M. Gratch and G. F. DeJong, "On comparing operationality and utility," Technical Report UIUCDCS–R–91–1713, Department of Computer Science, University of Illinois, Urbana, IL, 1991. (Submitted to Machine Learning Journal)

[Greiner92] R. Greiner and W. W. Cohen, "Probabilistic Hill–Climbing," *Proceedings of Computational Learning Theory and 'Natural' Learning Systems*, 1992. ((to appear))

[Laird86] J. E. Laird, P. S. Rosenbloom and A. Newell, *Universal Subgoaling and Chunking: The Automatic Generation and Learning of Goal Hierarchies*, Kluwer Academic Publishers, Hingham, MA, 1986.

[Leckie91] C. Leckie and I. Zukerman, "Learning Search Control Rules for Planning: An Inductive Approach," *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, IL, June 1991, pp. 422–426.

[Letovsky90] S. Letovsky, "Operationality Criteria for Recursive Predicates," *Proceedings of the National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 936–941.

[Markovitch89] S. Markovitch and P. D. Scott, "Utilization Filtering: a method for reducing the inherent harmfulness of deductively learned knowledge," *Proceedings of The Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989, pp. 738–743.

[Minton85] S. Minton, "Selectively Generalizing Plans for Problem–Solving," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, August 1985, pp. 596–599.

[Minton88] S. N. Minton, "Learning Effective Search Control Knowledge: An Explanation–Based Approach," Ph.D. Thesis, Department of Computer Science, Carnegie–Mellon University, Pittsburgh, PA, March 1988. (Also appears as CMU–CS–88–133)

[Mitchell83] T. M. Mitchell, P. E. Utgoff and R. Banerji, "Learning by Experimentation: Acquiring and Refining Problem–solving Heuristics," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, T. M. Mitchell (ed.), Tioga Publishing Company, Palo Alto, CA, 1983, pp. 163–190.

[Mitchell86] T. M. Mitchell, R. Keller and S. Kedar–Cabelli, "Explanation–Based Generalization: A Unifying View," *Machine Learning 1*, 1 (January 1986), pp. 47–80.

[Nadas69] A. Nadas, "An extension of a theorem of Chow and Robbins on sequential confidence intervals for the mean," *The Annals of Mathematical Statistics 40*, 2 (1969), pp. 667–671.

[Natarajan89] B. K. Natarajan, "On Learning from Exercises," *Proceedings of the Second Annual Workshop on Computational Learning Theory*, Santa Cruz, CA, JULY 1989, pp. 72–87.

[Richards91] B. L. Richards and R. J. Mooney, "First–order theory revision," *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, IL, June 1991, pp. 447–451.

[Ruby91] D. Ruby and D. Kibler, "SteppingStone: an empirical and analytical evaluation," *Proceedings of the National Conference on Artificial Intelligence*, Anaheim, CA, July 1991, pp. 527–532.

[Shavlik88] J. W. Shavlik, "Generalizing the Structure of Explanations in Explanation–Based Learning," Ph.D. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, January 1988. (Also appears as UILU–ENG–87–2276, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana–Champaign.)

[Smith85] D. E. Smith and M. R. Genesereth, "Ordering Conjunctive Queries," *Artificial Intelligence 26*, 2 (1985), pp. 171–215.

[Subramanian90] D. Subramanian and R. Feldman, "The Utility of EBL in Recursive Domain Theories," *Proceedings of the National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 942–949.

[Tadepalli91] P. Tadepalli, "Learning with Inscrutable Theories," *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, IL, June 1991, pp. 544–548.

[Towell90] G. G. Towell, J. W. Shavlik and M. O. Noordewier, "Refinement of approximate domain theories by knowledge–base neural networks," *Proceedings of the National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 861–866.

[Utgoff91] P. E. Utgoff and J. A. Clouse, "Two kinds of training information for evaluation function learning," *Proceedings of the National Conference on Artificial Intelligence*, Anaheim, CA, July 1991, pp. 596–600.

[Wellman90] M. P. Wellman, *Formulation of Tradeoffs in Planning Under Uncertainty*, Pitman and Morgan Kaufmann, 1990.

[Wilkins89] D. C. Wilkins and Y. Ma, "Sociopathic knowledge bases: correct knowledge can be harmful even given unlimited computation," Technical Report UIUCDCS–R–89–1538, Department of Computer Science, University of Illinois, Urbana, IL, 1989.

[Woodroofe82] M. Woodroofe, *Nonlinear Renewal Theory in Sequential Analysis*, SOCIETY for INDUSTRIAL and APPLIED MATHEMATICS, Philadelphia, PA, 1982.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-92-1723	2.	3. Recipient's Accession No.
4. Title and Subtitle An Analysis of Learning to Plan as a Search Problem			5. Report Date January 1992 6.
7. Author(s) Jonathan Gratch and Gerald DeJong			8. Performing Organization Rept. No. R-92-1723
9. Performing Organization Name and Address Department of Computer Science 1304 W. Springfield Avenue Urbana, IL 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No. NSF IRI 87-19766
12. Sponsoring Organization Name and Address National Science Foundation			13. Type of Report & Period Covered Technical 14.

15. Supplementary Notes

16. Abstracts

Increasingly, machine learning is entertained as a mechanism for improving the efficiency of planning systems. Research in this area has generated an impressive battery of techniques and a growing body of empirical successes. Unfortunately the formal properties of these systems are not well understood. This is highlighted by a growing corpus of demonstrations where learning actually *degrades* planning performance. In this paper we view learning to plan as a search problem. Learning is seen as a transformational process where a planner is tailored to a particular domain and problem distribution. To accomplish this task, learning systems draw from a vocabulary of transformation operators such as macro–operators or control rules. These "learning operators" define a space of possible transformations through which a system must search for a efficient planner. We show that the complexity of this search precludes a general solution and can only be approached via simplifications. We illustrate the frequently unarticulated commitments which underly current learning approaches. These simplifications improve learning efficiency but not without tradeoffs. In some cases these tradeoffs result in less than optimal behavior. In others, they produce planners which become worse through learning. It is hoped that by articulating these commitments we can better understand their ramifications. Finally, we discusses a particular learning technique which explicitly utilizes these simplifications to ensures performance improvements with reasonable efficiency.

17. Key Words and Document Analysis. 17a. Descriptors

Learning and Planning, Empirical methods, Theoritical and Empirical evaluation

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
unlimited	UNCLASSIFIED	17
	20. Security Class (This Page) UNCLASSIFIED	22. Price