Appears in Ninth International Machine Learning Conference, Aberdeen, Scotland, 1992, pp. 178–188

An Analysis of Learning to Plan as a Search Problem

Jonathan Gratch and Gerald DeJong
Beckman Institute for Advanced Studies,University of Illinois
405 N. Mathews, Urbana, IL 61801
e–mail: gratch@cs.uiuc.edu

Abstract

COMPOSER is one of a growing number of techniques for learning to plan. Like other approaches, it embodies a number of simplifications to overcome the complexities of learning. These simplifications introduce tradeoffs between learning efficiency and effectiveness. In this paper we relate COMPOSER to our general framework of simplifications for learning to plan [Gratch92a]. This discussion illustrates how such a framework may be used to analyze a particular approach, highlighting the learning system's strengths and weaknesses.

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, Segre88], 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 approaches are not well understood. This is highlighted by demonstrations where learning *degrades* planning performance [Etzioni90b, Gratch91a, Minton85, Subramanian90].

In this paper we relate a particular learning system to our general framework for learning to plan described in [Gratch92a] The framework provides a unifying perspective where seemingly different approaches are related through their use of common simplifying assumptions. We review the framework (Section 3) and then turn to describing the COMPOSER [Gratch92b] technique from this new perspective (Section 4). The discussion illustrates the framework's use as an analysis tool. It also provides a detailed illustration of the tradeoffs involved in some common learning simplifications. The next section motivates the view of learning to plan as a search problem.

2 LEARNING AS SEARCH

In classification learning, techniques are frequently characterized as search processes (see [Mitchell82]). Some novel issues arise when this view is extended to techniques for learning to plan. In this context, a learning system takes a particular planner operating in a particular domain, and tailors it to more effectively solve problems. From the search perspective, this can be viewed as a transformational process where a series of transformations are applied to the original problem solver. A planner may be transformed in a variety of ways. They include the introduction of search control knowledge [Braverman88, Mitchell83] and refinements to a world model [Richards91, Towell90].

Each learning technique utilizes a *vocabulary of transformations*. These are "learning operators" and collectively they define a *transformation space*. For instance, acquiring a macro–operator can be viewed as an operation which transforms the initial system (the original planner) into a new system (the planner operating with the macro–operator). A learning technique must explore this space of potential transformations for a sequence which results in a more effective planning system.

The quality of a learning technique can be characterized along two dimensions: the efficiency of its search and the effectiveness of the transformed planner. Efficiency can be evaluated by complexity measures. The effectiveness of a planner depends on the objectives of the agent which uses it. Common measures of effectiveness include average planning time [Minton88], probability of goal satisfaction [Drummond90], and solution quality [Eskey90]. These measures are usually dependent on the expected distribution of planning problems present in the task environment.

In [Gratch92a] we draw on these notions of efficiency and effectiveness to define a particular class of learning techniques called *minimally adequate learners*. Given an initial planner, a minimally adequate technique applies zero or more transformations, resulting in a planner of equal or (with high probability) higher effectiveness. Furthermore, the learning process must be tractable. This definition does not require that each transformation monotonically improve planning performance. Rather the learning system must terminate with something equal or better than the initial planner. A surprising observation is that many learning to plan techniques do not insure this minimal requirement [Gratch91b, Minton85, Subramanian90].

We assume that effectiveness is characterizable by a distribution sensitive *utility function*. For example, if the objective is to reduce problem solving cost, utility could be related to the sum of the solution cost of each problem in the distribution weighted by its probability of occurrence:

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

Utility is a preference function over planners. It is also useful to discuss the utility of individual transformations. The *incremental utility* of a transformation is defined as the change in utility that results from applying the transformation to a particular planner (e.g. adopting a control rule). This means the incremental utility of a transformation is conditional on the planner to which it is applied. We denote this as: ΔUTILITY(Transformation|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 many learning systems do not explicitly evaluate utility. For example, many speed–up learning systems incorporate *operationality criteria* to determine which transformations to adopt (e.g. [Braverman88, Letovsky90, Mitchell83]). This can be viewed as a binary approximation to incremental utility — transformations which satisfy the operationality criteria are expected to exhibit positive incremental utility, while those rejected by the criteria are expected to exhibit negative incremental utility.

From the search perspective, a learning system must explore the transformation space to identify preferred planners. This task can be decomposed into three basic components:

* A search component that explores the space.

* A utility estimation component which determines which transformations to incorporate into the final planner,

* An observation component which provides information about the task environment and the performance of the planner to the other two components.

The search space is determined by which transformations are proposed by the learning system and what combinations of transformations are sanctioned. Typically, transformations are proposed in response to problem–solving successes or impasses. Incremental utility might be estimated by combining information from multiple problems drawn from the distribution. The observation component must extract from examples the particular information required to propose and evaluate transformations.

3 FRAMEWORK OF SIMPLIFICATIONS

There are three challenges to adequate learning: 1) the space of possible transformations may be large, 2) it is difficult to reliably estimate effective transformations, and 3) it may be expensive to extract the necessary information from the task environment. Taken together, these difficulties suggest there there does not exist a general solution to the problem of adequate learning. Nevertheless, there are a number of published techniques which claim to work well. This apparent contradiction is resolved by noting that learning techniques adopt simplifications to address each challenge. Many of these simplifications embody a simple tradeoff: efficiency is gained by adopting less than optimal transformation sequences. Other simplifications are heuristic and can result in inadequate behavior under some (difficult to articulate) circumstances.

In this section we outline the basic simplifications adopted by learning to plan techniques (a more complete treatment appears in [Gratch92a]). Simplifications are rarely explicitly noted in these works. It is often difficult to determine the precise simplifications an algorithm embodies. We present concise and obvious simplifications to each challenge. We then argue how different techniques are best viewed as approximating these commitments. The framework is representative of the approaches which are popular in the literature rather than an exhaustive list of possible approaches. The discussion is organized into approaches for each of the three basic challenges.

3.1 TRANSFORMATION SPACE COMPLEXITY

A learning system must explore a potentially large space of transformation sequences. This space is infinite if the transformation vocabulary is unbounded or if the same transformation can be applied an unbounded number of times. Even with a finite vocabulary and restricting each transformation to at most one application, the complexity is still daunting: for n transformations there are $O(n!)$ distinguishable sequences (any planner may be the result of up to n ordered transformations). Depending on the vocabulary of transformations and utility function, each of these alternative planners can have different utility.

3.1.1 Utility Equivalence Classes

Under some circumstances the complexity of the transformation space is misleading. For example, depending on the vocabulary of transformations, the order in which transformations are adopted may be irrelevant to the utility of the final planner. In this circumstance, any of the $m!$ permutations of a set of m transformations. would yield planners with equivalent utility if applied to the initial planner. These permutations form a *utility equivalence class*. If utility equivalence classes can be identified, they can be exploited to reduce search. For example, the learning system can narrow the search such that only one member of each equivalence class is considered.

Many learning systems adopt a particular simplification which is best viewed as exploiting an utility equivalence class. The simplification is to treat the incremental utility of a transformation as *independent* of other transformations. This means that the incremental utility of a transformation is the same regardless of what other transformations have been adopted. With this simplification a learning system can reduce an exponential search to $O(n)$ (see [Gratch92a]). This simplification leads to adequate learning under certain sufficient conditions. If the transformations are in fact independent, a learning system can in $O(n)$ identify the planner with *optimal* utility. If independence does not hold, this simplification can sacrifice minimal adequacy. For example, we document how negative interactions between control rules cause PRODIGY/EBL [Minton88] to to acquire harmful control strategies.

Many learning systems do not explicitly consider interactions (including SOAR [Laird86], STATIC [Etzioni90b], PRODIGY/EBL, RECEBG [Letovsky90], IMEX [braverman88], and PEBL [Eskey90]). Ma and Wilkins illustrate a similar situation for knowledge–base revision systems [Wilkins89]. Systems which do not adopt this simplification include PALO [greiner92], COMPOSER [Gratch92b], and [Leckie91].

3.1.2 Generation Pruning

Most learning systems employ powerful pruning techniques to reduce the space of alternatives. One way to reduce complexity is by restricting which transformations are actively considered. Most transformation vocabularies define a vast space of possible transformations. For example, a system using macro–operators might consider macros built from any legal sequence of operators in the domain. Most learning systems only consider a tiny fraction of the legal transformations. We say a system employs *generation pruning* if it restricts the class of transformations which are actively considered. A common approach is *event driven learning*. Under this strategy, transformations are only proposed in response to planning events such as success or failure which are observed in the course of problem solving. Etzioni's *nonrecursive hypothesis* can also be viewed as a generation pruning strategy [Etzioni90b]. His criterion states that a control rule should only be considered if it is based on a nonrecursive explanation. Rosenbloom, Lee, and Unruh provide an interesting discussion of the relationship between a planner's architecture and generation pruning [Rosenbloom92].

3.1.3 Composition Pruning

In addition to restricting the class of available transformations, many learning systems restrict how transformations are composed. For example, even when transformations are not order independent, many techniques only consider a single permutation of the transformation sequence, effectively pruning the other ordered sequences from the transformation space. In some cases the order is resolved in a particular way. In the case of macro–operators, Shavlik suggests an 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]. Thus the organization of macro–operators is determined by the order in which the transformations were adopted and the learning system cannot alter this order.

Another powerful simplification is to adopt a heuristic search technique like hill–climbing or beam search. For example, COMPOSER and PALO employ hill–climbing search to restrict the space of alternatives. A greedy technique can climb the gradient of incremental utility values, picking the transformation with highest incremental utility with respect to the previously selected transformation.

Pruning simplifications introduce tradeoffs. A 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 minimal adequacy, as it does not effect the perceived utility of the remaining transformations. Every learning system we have analyzed implements some form of generation or composition pruning.

3.2 ESTIMATION COMPLEXITY

The utility of a transformation depends on information which is frequently unavailable such as the distribution of future problems. The natural approach is to estimate utility from training examples. The simplest approach is estimation by brute force. If there is a known finite set of problems of interest, the planning system might attempt them all, observing its behavior. After applying a transform, the planner could be rerun on the set. The difference between the runs is, by definition, the incremental utility of the transformation. This procedure identifies a single incremental utility value and the process must be repeated for each utility determination. As the set of problems may be large, unavailable, or infinite, and there may be many transformations from which to choose from, this approach is clearly impractical.

3.2.1 Learning Without Examples

Some learning approaches employ syntactic criteria to identify transformations with positive incremental utility. For example, syntactic operationality criteria are often used to discriminate between transformations with positive and negative utility [Braverman88, Hirsh88, Letovsky90]. In this sense, operationality can be seen as a two–valued approximation of incremental utility. The STATIC [Etzioni90b] and RECEBG [Letovsky90] systems utilize criteria based on the concept of recursive unwindings. A potential problem with these approaches is that they do not account for distribution information. Thus it can be quite difficult to devise a sufficiently general criterion. One interpretation of these criteria is that they produce heuristic estimates — they do not guarantee accurate estimates for every distribution, but they are sufficiently close on "typical" distribution. Unfortunately it is quite difficult to characterize the distributions which are acceptable to these techniques.

3.2.2 Unquantified Error

The empirical approach views utility as a random variable. Incremental utility values on individual problems represent data points; utility is estimated as the mean of a sample of problems. This approach yields a learning system which is probabilistically adequate. PRODIGY/EBL was perhaps the first system to average incremental utility values across many training problems. One simplification is to forbid reasoning about the accuracy of utility estimates. The average of a sample may differ from the true average. A transformation which is estimated to be good may in fact have negative utility. The likelihood of a large discrepancy may be minimized by drawing more examples, but techniques which do not reason about the confidence of their estimates have no way of assessing if sufficient examples have been taken. Approaches which adopt the simplification of *unquantified error* require the user to determine the number of training examples. If this number is insufficient, the learning approach may not be minimally adequate.

3.2.3 Quantified Error

More recent approaches have provided bounds on the probability of mistakes. Greiner and Cohen [Greiner92] introduce a method based on Chernoff bounds. This distribution–free approach adopts transformations after drawing enough examples to ensure (with high probability) that the transformation will improve performance. 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 a number of examples polynomial in the error level.

3.3 OBSERVATION COMPLEXITY

Learning techniques must extract the necessary information to generate transformations and to estimate utility. This process is complicated by the fact that the form of the information depends on the utility function. A utility function based on planning cost requires information on how a transformation affects the resource usage patterns of the planner. A utility function based on solution quality requires first that a solution be generated, and second that information about quality be assessed. A technique which supports arbitrary utility functions is unlikely. Even with a fixed utility function, the problem of gathering information is often non–trivial. Ideally we have access to an efficient analytic model of the planner which can predict incremental utility. Unfortunately it is difficult to provide such a model for non–trivial planning systems. The common alternative is to to rely on empirical approaches which directly observe the planner in the course of problem solving. But empirical approaches introduce new difficulties. Planning is undecidable in general. There are obvious difficulties in measuring the performance of an intractable process.

3.3.1 Learning From Self–solutions

Many learning techniques use complete planning solutions to generate utility estimates. As the system is generating its own information, we refer to this process as *learning from self–solutions*. This approach equates observation complexity with the complexity of the (possibly transformed) 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, Gratch91b, Greiner92, Leckie91, Minton88, Mitchell83, Ruby91]. One might ask why we need to learn at all if problem solving is already feasible. However, in many circumstances this is quite reasonable. For instance, in situations where large numbers of problems must be solved, small increases in efficiency can result in huge savings.

3.3.2 Learning From Partial Solutions

The simplest approach to learning from self–solutions is to solve the same problem many times, comparing the difference between transformed and untransformed planners. However, as planning can be expensive, we may not have the resources to completely solve each problem. Considerable savings could result if the system can learn from partial solution attempts. Two obvious ways to generate partial solutions would be to terminate problem solving after the first sign of inefficiency (e.g. when backtracking occurs) or by giving the planner fewer resources during learning than are normally available.

Several learning techniques use information from partial solution attempts when they *generate* transformations. For example, SOAR interleaves generation and planning by proposing transformations after each planning impasse. In fact, most failure–driven learning techniques should be able to generate transformations in response to partial planning traces. Some approaches also extract utility information from partial solution traces. For example, in PRODIGY/EBL, an incomplete solution may contain several examples of control rule behavior. PRODIGY/EBL also incorporates a simplification where part of the utility estimate is derived from a single example. Unfortunately, extracting utility information from partial solution traces will only be effective if incremental utility is relatively homogeneous for the transformations in use. This sufficient condition may prove difficult to demonstrate. It it does not hold, the increase in observation efficiency is gained at the cost of estimation accuracy which in turn can violate minimal adequacy.

3.3.3 Learning From a Teacher

An alternative to self–solutions is to require a teacher to provide the appropriate data. This places observation complexity in the hands of the teacher. The teacher, with its superior knowledge, presumably can elicit the information with reasonable cost. Implementations of this approach place strong constraints on the information the teacher can provide. For example, Tadepalli introduces a system where the teacher is required to generate solutions in a particular, non–intuitive form [Tadepalli91]. Natarajan discusses another approach where the teacher must provide the system with optimal solution paths [Natarajan89]. The availability of a teacher can greatly simplify the observation complexity.

Unfortunately, as the necessary information is a function of the utility function and the vocabulary of transformations, it may be difficult to provide an adequate teacher.

3.3.4 Learning From Simpler Problems

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 leads to sophisticated problem solving ability. Minton informally adopts this approach in his evaluation of PRODIGY/EBL [Minton88 pp. 137–138]. In these experiments the training set is biased such that problem difficulty is gradually increased as learning proceeds. This idea shares the potential drawbacks of learning from partial information. The incremental utility of transformations varies across different problems. If the incremental utility of a transformation varies systematically with problem difficulty, a multi–example estimate of incremental utility will be compromised. Such an approach amounts to making an assumption that incremental utility is relatively insensitive to problem difficulty. Under certain circumstances, however, such biasing can prove effective. Natarajan demonstrates some sufficient conditions under which this type of learning is possible [Natarajan89].

3.3.5 Simultaneous Extraction

The previous simplifications address the complexity of solving a single problem. However, one solution attempt may be insufficient to determine the utility of a set of potential transformations. A brute force approach to estimate the utility of *n* transformations would be to solve the same problem *n+1* times (once without any transformation, and once with each of the *n* candidates). Incremental utility can be extracted by the difference in utility between runs. Many learning approaches implement observation procedures which enable the the extraction of utility values for multiple transformations from a single solution trace. We refer to such an approach as *simultaneous extraction*. PRODIGY/EBL, COMPOSER, SYLLOG [Markovitch89], and PALO [Greiner92] perform simultaneous extraction.

Simultaneous extraction can reduce the cost of extracting information. However, it is possible that such techniques might compromise the veracity of the utility values. To maintain accuracy, it must be the case that the incremental utility value determined for one transformation is not influenced by whatever other transformations are being determined. This condition does not always hold. For example, PRODIGY/EBL gathers statistics for control rules as other rules are learned and forgotten. These shifting conditions influence the estimates. In [Gratch91a] we illustrate a domain where these influences lead to learning behavior which is not minimally adequate.

4 APPLYING THE FRAMEWORK

There are a variety of simplifications to address the challenges of adequate learning. Each involves its own set of tradeoffs and in many cases these are difficult to evaluate. In this section we apply the framework to a particular learning technique. By grounding the framework in a particular system we can elaborate the consequence of some common simplifications. More importantly this section illustrates how the framework clarifies and organizes the process of analyzing a learning system. We characterize the COMPOSER system [Gratch92b]. The algorithm is summarized in Figure 2

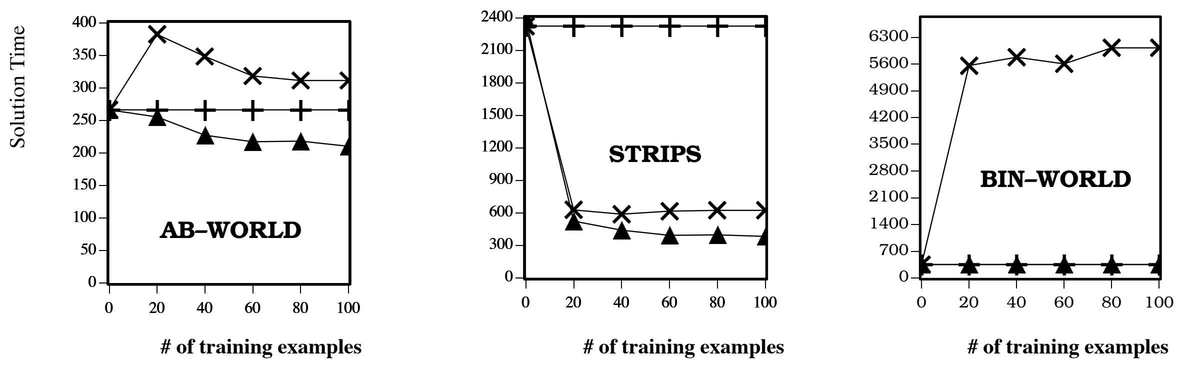
```
Input: TRAINING_EXAMPLES

CONTROL_STRATEGY = ∅
CANDIDATE_SET = ∅
While more training examples
  solve problem with Planner+CONTROL_STRATEGY
  learn new rules and add them to CANDIDATE_SET
  acquire statistics for all rules in CANDIDATE_SET from trace
  POSITIVE_RULES = ∅
  Forall rules ∈ CANDIDATE_SET
    If ΔUTILITY(rule|PRODIGY+CONTROL_STRATEGY)
      significantly negative
      Then remove rule from CANDIDATE_SET
    If ΔUTILITY(rule|PRODIGY+CONTROL_STRATEGY)
      significantly positive
      Then add rule to POSITIVE_RULES
  If POSITIVE_RULES
    add rule with highest utility to CONTROL_STRATEGY
    remove this rule from CANDIDATE_SET
    discard all statistics on rules in CANDIDATE_SET

Output: CONTROL_STRATEGY
```

Figure 2: The COMPOSER algorithm

COMPOSER learns search control rules to improve the average planning speed of a STRIPS–like planner. It can be viewed as a rigorous version of the utility analysis method introduced by PRODIGY/EBL [Minton88]. In fact, it is implemented with the PRODIGY 2.0 architecture. Its design was motivated by the observation that PRODIGY/EBL is not minimally adequate. Another paper elaborates these deficiencies which are shared by many other learning to plan techniques [Gratch91b]. In terms of our framework, COMPOSER can be viewed as relaxing some unjustified simplifications imposed by PRODIGY/EBL. In particular, COMPOSER eliminates an unjustified independence simplification in the search component, introduces an estimation technique with quantified error, and eliminates an unjustified partial solution simplification for extracting utility estimates. A number of other simplifications are retained and these are discussed at length.

4.1 EMPIRICAL EVALUATION

Before preceding, we briefly summarize an empirical evaluation of the COMPOSER approach from [Gratch92b]. We draw on these results in the later discussion. We tested the STRIPS domain from [Minton88], and two domains for which PRODIGY/EBL learns harmful control knowledge:



SYSTEM	AB–WORLD			STRIPS			BIN–WORLD		
	Rules Added	Train Time	Test Time	Rules Added	Train Time	Test Time	Rules Added	Train Time	Test Time
+ No Learning	——	——	266	——	——	2323	——	——	346
▲ COMPOSER	1	1667	210	4	4133	382	0	3425	346
✕ PRODIGY/EBL	11	1253	311	20	3775	622	2	6383	6020

Figure 3: Summary of empirical results

the AB–WORLD domain from [Etzioni90a] and the BIN–WORLD domain from [Gratch91a]. COMPOSER is compared against PRODIGY/EBL and a non–learning system. The results are summarized in Figure 3. In each domain the systems are trained on 100 training examples drawn randomly from a fixed distribution. The current control strategy is saved after every twenty training examples. The graphs illustrate learning curves where the independent measure is the number of training examples and the dependent measure is execution time for 100 test problems drawn from the same distribution. This process is repeated eight times, using different but identically distributed training and test sets. Values in Figure 3 represent the average of these eight trials. "Rules Added" indicates the average number of rules learned by the system; "Train Time" is the number of seconds required to process the 100 training examples (including the time to solve training problems, generate control rules, and perform utility analysis); "Test Time" is the number of seconds required to generate solutions for the 100 test problems. COMPOSER uses an error parameter which bounds the probability of adding a harmful control rule. This was set at 10%.

COMPOSER exhibits minimal adequacy on each domain while PRODIGY/EBL does not. In BIN–WORLD COMPOSER terminated without adopting any transformations (there does not appear to be a good control rule for this domain). Both systems have comparable efficiency.

4.2 TRANSFORMATION SPACE COMPLEXITY

COMPOSER combines both generation and composition pruning methods to address the complexity of the transformation space. COMPOSER uses PRODIGY/EBL's rule generator and thus inherits its generation pruning technique. After solving a problem, the rule generator proposes control rules based on observed planning inefficiencies. Different inefficiencies can arise in different problems. Furthermore, as control rules are adopted, old inefficiencies are corrected and new inefficiencies arise. In this sense the rule generator is a function which takes a particular problem and a particular transformation sequence, and generates a set of control rules. This bias is dynamic. It changes as the planner is transformed (see [Rosenbloom92]). This provides a strong (but difficult to analyze) bias on which of the syntactically valid control rules will be entertained in the transformation space. Note that this bias is influenced by a number of factors including which training examples are presented and the order in which transformations are adopted.

COMPOSER implements composition pruning through a greedy hill–climbing approach. As control rules are generated they are placed on a candidate set and statistically evaluated. The system adopts the first control rule which demonstrates positive incremental utility to a pre–specified confidence level. Recall that by definition, incremental utility is conditional on a particular planner. In this case incremental utility is estimated with respect to the planner using the previously adopted sequence of control rules. As rules are adopted they are placed at the end of this sequence. As utility is evaluated conditional on the current control strategy, this approach avoids the negative interactions between control rules which hampers PRODIGY/EBL's form of utility analysis. However, a disadvantage is that the system cannot exploit positive interactions between rules. For example, it may be the case that two control rules have negative incremental utility in isolation, but combine synergistically to produce a good control strategy. Hill–climbing techniques like COMPOSER cannot recognize such situations and can be caught on local maxima.

Collectively the generation and composition simplifications form a very strong bias. For example, in the STRIPS domain the number of syntactically valid control rules is effectively infinite. For our distribution of problems the rule generator produces fifty–seven distinguishable rules. This defines approximately 10^{76} distinguishable transformation sequences. The bias, however, enable COMPOSER to identify beneficial control strategies with only 165 incremental utility determinations on average. For AB–WORLD the system explores a potential space of 10^{30} transformation sequences with 119 utility determinations. In BIN–WORLD a space of 154 sequences is evaluated in five determinations.

Unfortunately, it is difficult to evaluate what this bias costs the system in terms of lost learning opportunities. Clearly, many transformation sequences are excluded from consideration. Many of these sequences could result in effective planners. The obvious to evaluating the potential loss is to determine the topology of the unbiased space can compare this with the biased sub–space. Unfortunately, determining utility values for 10^{76} strategies is impractical. However, we can make some statements based on an analysis of COMPOSER's transformation vocabulary and some less exhaustive empirical investigations.

The generation bias is difficult to analyze, but it seems unlikely that beneficial rules are being excluded. Recall that generation is biased by observed planning inefficiencies. An interesting question is what additional tradeoffs are embodied in the hill–climbing approach. First we note that local maxima cannot arise if the incremental utility of a transformation is independent of what other transformation have already been adopted. Under this circumstance the transformation space exhibits a single peak. If there are interactions, COMPOSER retains minimal adequacy but it can be prevented from finding optimal strategies.

Unfortunately, control rules can exhibit significant interactions (see [Gratch91b]). Furthermore, we have observed instances where COMPOSER adopts one transformation sequence where sequences with higher utility exist. For example, in [Gratch92b] we demonstrate the alternative hill–climbing approach embodied in PALO [Greiner92] can produce somewhat better control strategies, although at a substantial loss in efficiency. In summary, COMPOSER's biases enable it to identify better planners but it is susceptible to local maxima. Perhaps further analysis can lead to a result like greedy set covering where the discrepancy between global and local optimal is bounded. Short of this, it is difficult to assess the ultimate cost of this form of bias.

4.3 ESTIMATION COMPLEXITY

COMPOSER implements a statistical estimation technique which provides bounds on the probability of adopting harmful control rules. The incremental utility of a transformation is estimated by averaging utility values from successive, randomly drawn, problems. This is treated as a *sequential analysis* problem (see [Govindarajulu81]). Observations are gathered until a specified confidence level is reached. We use a distribution–free test developed by Nádas [Nadas69]. Statistics are gathered on each candidate control rule until the following inequality holds:

$$(V_{r,n}/\overline{X}_{r,n})^2 < n(1/a)^2$$

where $\overline{X}_{r,n}$ is the average utility of the rule r over n problems, ${V_{r,n}}^2$ is the sample variance, n is greater than or equal to three, and a is the $(1 - \delta)/2$th quantile of the standard normal distribution (see [Gratch92b] for more details).

After processing an example, the inequality is evaluated for each candidate rule. Any candidates which satisfy the inequality and have negative incremental utility ($X_{r,i} < 0$) are removed from the candidate set. If candidates satisfy the inequality and have positive incremental utility ($X_{r,i} < 0$), the rule with highest incremental utility is added to the transformation sequence. If a transformation is adopted, the statistics for the remaining candidate rules are discarded as they reflect the previously transformed planner (recall that incremental utility is conditional on a particular planner).

Statistical models necessarily introduce simplifications into the estimation process. Although techniques are often described as "distribution–free," they make weak assumptions about properties of the distribution. For example, one minimally requires that the distribution of utility values has finite variance. In recent years, statistical work in machine learning has been influenced by the weak statistical models favored in computational learning theory [Valiant84]. These models are based on worst–case analysis and frequently provide overly–conservative estimates of error (see [Buntine89] for a critique of this model). Conservative estimates adversely impact learning efficiency as they require more examples than necessary to achieve significance.

Practical experience in the field of statistics has demonstrated that stronger simplifications can be reasonably adopted. The success of modern statistics is based on the Central Limit Theorem [Hogg78 p. 192] which demonstrates that the distribution of the mean of a sample tends to a normal distribution as the size of the sample grows. A traditional simplification is to treat the mean as if it is normally distributed.[1] COMPOSER's estimation technique is based on the Central Limit Theorem. This simplification implies that if an error level of δ is specified, the observed error rate will be *approximately* δ; it may be somewhat higher or lower. The discrepancy is a function of the underlying utility distribution. Practical experience has validated this approximation in practice.

The empirical evaluation of COMPOSER allows us to test the reasonableness of our statistical simplifications. For each domain COMPOSER was tested on eight independent learning trials for a total of twenty–four trials. The error rate was set at 10%. Across these trials the system adopted forty–three transformations. Thirty–nine of these transformations improved planning utility. Four transformations hurt

1. Actually, the common procedure is to approximate the distribution of the mean with a distribution that converges to normal. This is the rationale behind the t–test.

performance and can be characterized as mistakes (STRIPS – 36 transformations, 4 errors; AB–WORLD – 7 transformations, 0 errors; BIN–WORLD – 0 transformations, 0 errors). Collapsing across domains this yields an observed error rate of 11%. This difference from the intended level of 10% is not statistically significant. Figure 4 shows the learning curves for each trial in the STRIPS domain. Mistakes occurred on trials three, four, and eight. The graphs illustrate that when errors occurred, they tended to be small. This is expected as the statistical model predicts that the probability of an error diminishes with its severity.

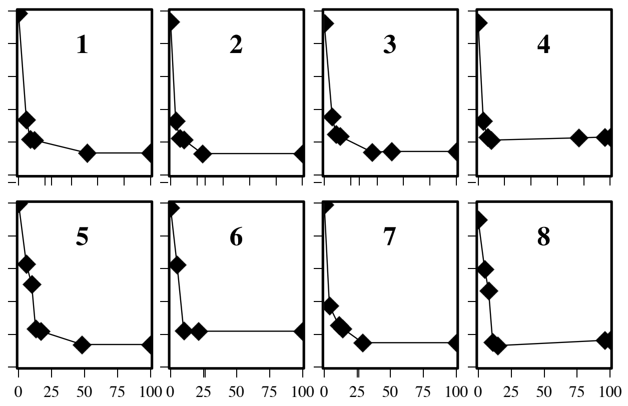


Figure 4: Individual trials for STRIPS domain

The approach also performs well in the number of training examples required for each utility estimate. Estimates averages fifteen examples to reach significance (minimum three, maximum eighty–five). This is in strong contrast to computational learning theory style bounds which require thousands of examples (see [Gratch92b]). A general trend was that transformations with higher incremental utility required fewer examples to reach significance.

4.4 OBSERVATION COMPLEXITY

Observation complexity is the challenge to which COMPOSER makes its most restrictive simplification. There does not exist a practical cost model for the PRODIGY planner. Thus, COMPOSER uses self–solutions to extract information for rule generation and evaluation. The control rule generator only needs partial solution traces to conjecture control rules. However, to extract incremental utility values, we insist on complete solution traces. This requirement is motivated in [Gratch91b]. The difficulty is that the utility of a control rule varies heterogeneously within a given solution so that partial information is misleading.

If *n* rules are in COMPOSER's candidate set, the simplest approach to extracting utility information is to solve each problem *n+1* times (see Section 3.3.5). COMPOSER implements a simultaneous extraction technique such that all *n* utility values can be extracted from a single solution trace. The technique requires matching the preconditions of control rules without actually adopting their control recommendations. After each solution, COMPOSER analyzes the annotated solution trace and extracts utility values for each candidate rule. The advantage of this approach is that it diminishes the number of solutions. The disadvantage is that it is more expensive to process each example. In addition to the normal solution cost, the planner pays the additional cost of evaluating candidate rule preconditions.

In principle, COMPOSER can accurately extract the estimates for an arbitrary number of candidate control rules. In practice, it is a difficult task to implement a system which accurately measures its own resource use. Empirically, COMPOSER exhibits good overall behavior. However, in viewing the system from our framework it occurred to us that simultaneous extraction could be tested directly.

Simultaneous extraction is justified if utility values derived for one candidate control rule are not influenced by the presence of other candidate rules. We tested this justification by comparing utility values of a rule in isolation against utility values of the same rule in the context of other candidate rules. For each domain, up to twenty control rules were selected at random and placed in the candidate set. Utility values were extracted over twenty planning problems. Each rule was then placed on the candidate set by itself and utility values were extracted over the same twenty problems. Finally, we derived the "true" incremental utility values by executing the planner *n+1* times (with no control rules, and with *n* strategies consisting of one rule each), again over the same twenty planning problems. Values across the twenty problems were averaged. This process was repeated many times for each domain. Figure 5 illustrates a one trial from the STRIPS domain. It illustrates a trend which was seen in every trial. The vertical axis is incremental utility (in seconds). The horizontal axis represents ten rules sorted by their true incremental utility.

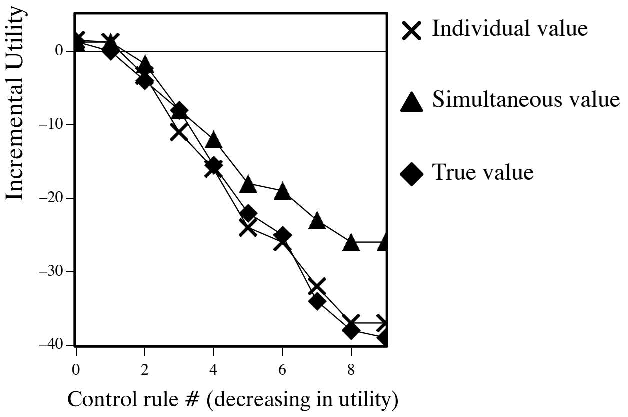


Figure 5: Simultaneous Extraction Test

The results indicate that utility value are accurate when a control rule is the only member of the candidate set. However, extracted values may differ when several other candidate rules are be estimated simultaneously. The trend on all of our tests is the same: utility is *overestimated* for rules with negative incremental utility. The discrepancy increases for rules with larger negative utility. Essentially our algorithm is undercounting the match cost for harmful control rules when many rules are on the candidate set. We have not observed a case where a rule with negative utility is assigned positive utility values, thus this effect does not compromise minimal adequacy in practice, but it does reveal a problem with our implementation.

This experiment is not interesting from a theoretical perspective as it simply indicates an error in our implementation. However, it illustrates how our framework highlights the type of issues which are often taken for granted. Standard experimental evaluations treat a learning system as a large black box. Our frameworks decomposes systems into many specific design decisions. Experiments like this one amount to validating the assumptions which underly a learning technique. They, therefore, provide greater information on why a technique does or does not work.

COMPOSER's observation complexity reflects two costs: the cost of solving problems, and the overhead of evaluating candidate rules. Problems are solved using the currently transformed planner. Thus, solutions are produced more efficiently as transformations are adopted. The overhead cost is minimized by discarding rules from the candidate set if they exhibit negative incremental utility. The results in Figure 3 indicate that COMPOSER's observational complexity compares favorably with the training times for PRODIGY/EBL. Thus, at least in these domains, COMPOSER has achieved minimal adequacy (an advance over PRODIGY/EBL) without a large reduction in efficiency. COMPOSER actually exhibits higher efficiency in the BIN–WORLD domain. However, while this is an advance over PRODIGY/EBL, it is clear the neither technique is appropriate when self–solutions are infeasible to generate. The common justification for this type of simplification (which is very common in learning to plan techniques) is that a high learning cost can be amortized over many test problems.

To summarize, COMPOSER implements a number of simplifications to address the complexities of learning to plan. The space of transformations is reduced by applying strong generation and composition pruning techniques including event driven learning and hill–climbing. Incremental utility is estimated by random sampling across the problem distribution. Errors our bounded with a statistical technique. Information for rule generation and estimation is obtained by simultaneous extraction over self–solutions. The technique achieves minimal adequacy with high probability. The disadvantage is that the technique may terminate at local maxima, the error rate may be higher than expected over some distributions, and the technique is restricted to problem classes where it is feasible to generate self–solutions.

5 CONCLUSIONS

Learning to plan embodies a number of complexities which preclude a general solution. Techniques are confronted with a vast space of alternatives. Even estimating beneficial transitions in this space can involve considerable expense. Approaches address this complexity by adopting simplifying assumptions. These simplifications increase efficiency but they frequently introduce tradeoffs. Furthermore, as these simplifications are often implicit, it is difficult to evaluate the behavior of particular learning techniques.

We described the COMPOSER system from the more general perspective of a framework of simplifications for learning to plan. Such a perspective forces us, as authors of a system, to recognize a number of simplifications which are implicit in the architecture. Once these simplifications are made explicit, they can be evaluated objectively. Instead of experimentally validating an approach as a single "black box," the framework facilitates a finer grained analysis of different components of the system. In the case of COMPOSER, many of these simplifications are shared by other learning techniques. Thus, our evaluation can provide at least indirect evidence on the validity of assumptions in the context of these other systems.

Acknowledgements

This research is supported by the National Science Foundation, grant NSF IRI 87–19766. We thank Nick Lewins, Dan Oblinger, Russ Greiner, and David Page for many interesting comments and discussions. We extend special appreciation to the PRODIGY group at CMU who provided us with a copy of their system.

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.

[Buntine89] W. Buntine, "A Critique of the Valiant Model," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989, pp. 837–842.

[DeJong86] G. F. DeJong and R. J. Mooney, "Explanation–Based Learning: An Alternative View," *Machine Learning 1*, 2 (April 1986), pp. 145–176.

[Drummond90] M. Drummond and J. Bresina, "Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction," *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 138–144.

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

[Etzioni90b] O. Etzioni, "Why Prodigy/EBL Works," *Proceedings of the National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 916–922.

[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.

[Govindarajulu81] Z. Govindarajulu, *The Sequential Statistical Analysis*, American Sciences Press, INC., Columbus, OH, 1981.

[Gratch91a] 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.

[Gratch91b] 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.

[Gratch92a] J. Gratch and G. DeJong, "A Framework of Simplifications in Learning to Plan," *First International Conference on Artificial Intelligence Planning Systems*, College Park, MD, 1992.

[Gratch92b] J. Gratch and G. DeJong, "COMPOSER: A Probabilistic Solution to the Utility Problem in Speed–up Learning," *Proceedings of the National Conference on Artificial Intelligence*, San Jose, CA, July 1992.

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

[Hirsh88] H. Hirsh, "Reasoning about Operationality for Explanation–Based Learning," *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI, June 1988, pp. 214–220.

[Hogg78] R. V. Hogg and A. T. Craig, *Introduction to Mathematical Statistics*, Macmillan Publishing Co., Inc., London, 1978.

[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.

[Mitchell82] T. M. Mitchell, "Generalization as Search," *Artificial Intelligence 18*, 2 (1982), pp. 203–226.

[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.

[Rosenbloom92] P. S. Rosenbloom, S. Lee and A. Unruh, "Bias in Planning and Explanation–Based Learning,"in *Machine Learning: Induction, Analogy and Discovery*, S. Chipman, A. Meyrowitz (ed.), Kluwer Academic Publishers, Hingham, MA. In Press, 1992.

[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.

[Segre88] A. M. Segre, *Machine Learning of Robot Assembly Plans*, Kluwer Academic Publishers, Norwell, MA, March 1988.

[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.)

[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.

[Valiant84] L. G. Valiant, "A Theory of the Learnable," *Communications of the Association for Computing Machinery 27*, (1984), pp. 1134–1142.

[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.