Learning Search Control for Constraint-Based Scheduling

Megan Eskey and Monte Zweben
NASA Ames Research Center
M.S. 244-17
Moffett Field, California 94035
eskey@pluto.arc.nasa.gov, zweben@pluto.arc.nasa.gov

Abstract

This paper describes an application of an analytical learning technique, Plausible Explanation-Based Learning (PEBL), that dynamically acquires search control knowledge for a constraint-based scheduling system. In general, the efficiency of a scheduling system suffers because of resource contention among activities. Our system learns the general conditions under which *chronic* contention occurs and uses search control to avoid repeating mistakes. Because it is impossible to *prove* that a chronic contention will occur with only one example, traditional EBL techniques are insufficient. We extend classical EBL by adding an empirical component that creates search control rules only when the system gains enough confidence in the plausible explanations. This extension to EBL was driven by our observations about the behavior of our scheduling system when applied to the real-world problem of scheduling tasks for NASA Space Shuttle payload processing. We demonstrate the utility of this approach and provide experimental results.

Introduction

In many real-world scheduling domains, activities are highly constrained by their need to share a finite set of resources. Efficient allocation of these resources and efficient search strategies are imperative in any scheduling system. Recognizing and anticipating situations of likely resource contention and determining an appropriate search strategy to avoid backtracking can improve scheduling and resource allocation. To synthesize schedules with minimum work-in-process (WIP) time, activities are scheduled from a temporally-based perspective. In a constraint-based scheduling system, this means committing to an activity's first available start and end times consistent with all temporal constraints, then selecting a resource that is available in that time interval. However, if there is no resource of the appropriate type available at that time, new start and end times must be chosen, causing backtracking. Scheduling from a resource-based perspective means choosing an appropriate resource, then finding an activity's start and end times based on the capacity constraints of that resource (i.e., the next time the resource is available). This strategy reduces backtracking, but the efficiency improvement is gained at the expense of increased WIP time. Previous scheduling methods have been to perform a heuristic look-ahead to determine contentious resources, and to schedule only these "bottleneck resources" from the resource-based perspective [Smith and Ow, 1985, Fox, 1987]. Rather than relying *only* on opportunistic schemes, we exploit previous experience by learning the general conditions under which backtracking occurs. We use the learned search control knowledge in future situations to alter the search strategy accordingly.

Because of the exponential size of the search space, the ability to use search control knowledge has been shown to be critical in many domain-independent problem solvers, including planning, scheduling, and theorem-proving systems [Minton, 1988b, Laird *et al.*, 1986, Smith and Ow, 1985]. We use (and extend) existing explanation-based learning (EBL) methods to dynamically acquire search control rules. We have added an empirical component to EBL, in which multiple examples confirm plausible conjectures. Unlike most previous work that combines empirical techniques with EBL [Pazzani *et al.*, 1986, Mooney and Ourston, 1989, Hirsh, 1989, Danyluk, 1989], our empirical component is not used to make inductive leaps. Rather, it is used because of the inherent characteristics of the target concept; the target concept cannot be proved with a single example and requires a distribution of examples. This extension was driven by the application of EBL to the real-world problem of scheduling tasks for Space Shuttle payload processing. The target concept in this domain is a *chronic* resource contention and cannot be confirmed with one example. We call this extension to EBL "Plausible Explanation-Based Learning" (PEBL).

This paper discusses the integration of PEBL into a constraint-based scheduling system. After a brief description of the scheduling system and the Kennedy Space Center (KSC) payload processing domain, we describe PEBL and how it is used in the context of scheduling. We then present results of a set of experiments that reveal the advantages and disadvantages of using the learned search control rules.

Constraint-based Scheduling

In this section, we describe our formulation of a constraint-based scheduling problem, and the specific problem of payload processing.

Scheduling as a Constraint Satisfaction Problem

A constraint satisfaction problem (CSP) is characterized by a finite set of variables $V = \{V_1 ... V_n\}$, their respective domains $D_1 ... D_n$, and a set of constraints $C(V_i, V_j, ..)$. A solution to a CSP is an assignment of values x_i to all V_i such that all constraints are satisfied. A constraint is a subset of the Cartesian product $C \subseteq D_1 \times D_2 \times ... \times D_n$, that specifies the consistent and inconsistent choices among variable values. Constraints may be specified extensionally by enumerating all consistent values, or specified intensionally as functions.

A scheduling problem can be viewed as a constraint satisfaction problem. In our scheduling system, we formulate a resource allocation and scheduling problem as a set of task and resource objects [Zweben and Eskey, 1989]. Task objects contain information (slots) about an activity's start time, end time, duration, and resource requirements. This information is represented by distinct variables each having a domain of possible values and a set of domain constraints. Resource requirements of a task include information about type and quantity needed. Resource objects contain information about the availability of the resources. Constraints in the system define temporal relations and resource capacity requirements. We specify constraints intensionally.

The scheduling process begins by placing all tasks on a priority queue of unscheduled tasks. The priority queue generally prefers tasks that are close to the anchors of the schedule (the tasks fixed in time due to external forces). As the tasks are removed from the queue, a variable commitment strategy is used to guide the search for appropriate times and resources for the task. When scheduling from a temporally-based perspective, start time, end time and duration variables are instantiated before resource requirement variables. When scheduling from a resource-based perspective, resource requirement variables are instantiated before temporal variables. A solution to the scheduling problem is an assignment of times and resources to each task, such that all constraints are satisfied.

The Kennedy Space Center Payload Processing Domain

We have formulated a scheduling problem based on the NASA space shuttle's payload processing domain [Hankins *et al.*, 1985, Brown, 1987]. Payloads that fly on the shuttle rest on modular containers called carriers. Kennedy Space Center personnel have generated a partially ordered hierarchy of tasks necessary to process the payloads and carriers before and after a shuttle flight. Resource type and quantity requirements for each task have also been determined. The anchor of the schedule in this domain is the launch date of the shuttle. The task queue is ordered such that post-launch tasks are scheduled forward in chronological order (with respect to the partial order) and pre-launch tasks are scheduled backwards from launch in reverse chronological order.

In this domain there are a limited number of expensive, specialized resources (e.g., the carriers) and a number of other scarce resources. Each mission's processing tasks must be completed in time for the next mission that requires the specialized or scarce resource. Thus, if WIP time is too long, a mission's launch date could potentially slip. Additionally, one must quickly synthesize viable schedules since time is also a critical resource. Although scheduling from the resource perspective generally improves the efficiency of the system, if used without restraint it can degrade the optimality of the solution by creating schedules with unacceptably long WIP times. Thus, our goal is to restrict the changes in the search strategy to the most critical points in the scheduling process.

Plausible Explanation-based Learning

In this section we describe the learning technique used to improve search.

The Basic Learning Method

Explanation-based learning is an analytical technique that can be used to learn search control knowledge [Mitchell *et al.*, 1986, DeJong and Mooney, 1986, Minton, 1988a]. EBL begins with a high-level target concept and a training example. Then, using a domain theory (a set of axioms describing the domain), it generates an explanation of why the particular training example is an instance of the target concept. The explanation is a proof that the training example satisfies the target concept. The EBL approach finds the weakest preconditions of the explanation, producing a learned description that implies the target concept. The description is a generalization of the training example and a specialization of the target concept and must satisfy some operationality criterion that requires that it be in a usable form.

In the domain of scheduling, the resources that are in high demand are considered bottlenecks if their capacity is insufficient for the resource requirements of concurrent tasks. The tasks competing for the bottleneck resources vary depending on the specific conditions of the current schedule. In each instance of contention, it is unlikely that the exact configuration of tasks will compete for some resource more than once.

Given:

1. A domain theory
2. A target concept
3. An example instance
4. A library of previously generated explanations

1. Try to instantiate one of the previously generated explanations for the given example instance.
2. If no explanation applies to the example instance, then use standard EBL to explain the instance: using the domain theory, prove that the example instance entails the target concept and generalize the explanation. Go to 5.
3. If an explanation applies to the example instance, augment the confidence in the explanation.
4. If the stored explanation has reached its confidence level, (based on some decision function, $D_{useful-explan}$), then extract an operational concept from the explanation and form a search control rule.
5. Continue problem solving.

Figure 1: The PEBL Procedure

On the other hand, some subset of these tasks may reappear in a similar configuration. With standard EBL, one must be able to prove that the example entails the target concept, but proving that some resource contention is *chronic* requires multiple examples. We define our target concept in this domain as a "plausibly chronic resource bottleneck". Thus, each example of contention is only a plausible conjecture of chronic contention, and empirical confirmation with multiple examples is needed to prove the target concept. Only when the system has gained enough confidence in the plausible explanation is a search control rule created. The confidence in an explanation can be described as a decision function, $D_{useful-explan}$, where:

$$D_{useful-explan} : P_{explan} \times C_m \times C_o \longrightarrow \{0, 1\}.$$

P_{explan} is the probability that the explanation is correct, C_m is the expected cost to match the search control rule [Minton, 1988a], and C_o is a measure of the expected degradation in the quality of the solution. The semantics of the rules in the domain theory can be viewed as conditional probabilities. Then, the conditional probabilities associated with the *plausible* implications can be used to determine the probability that an explanation is correct over some sample space [Pearl, 1985]. The PEBL procedure is shown in Figure 1.

An Example in the Payload Processing Domain

In the payload processing domain, the target concepts to be learned are the plausibly chronic resource bottlenecks. The domain theory contains information describing the conditions under which contention might occur, such as proximity of multi-mission launch dates, types of carriers, types of tasks, and types of resources. The training example is an instance of a task whose resource requirements were not met. As an example, we illustrate the generalization problem for a plausibly chronic resource bottleneck between tasks in two different missions (see Table 1). The domain theory is used to create a specialized instance of the target concept (with each of the variables instantiated). Then a goal regression algorithm generalizes the description by regressing the target concept's preconditions back through the results of the inference process, and finds the weakest preconditions under which the explanation holds.

For example, consider a task *spacelab-systems-experiment-test* that is a subtask of *spacelab-experiment-integration. Spacelab-systems-experiment-test* is part of a biological mission that uses a long-module-2 carrier. Among its resource requirements is a request for automatic test equipment. If there is no automatic test equipment available at the first time consistent with its temporal constraints, learning would be initiated. First, all previously generated explanations are tested to see if their preconditions apply to this particular example. If no explanation holds, a new one is generated that describes the current situation. In this example, by using the domain theory, the system would determine that the bottleneck is a result of contention with a set of previously scheduled tasks in some other biological mission that use all of the automatic test equipment. Both missions have launch dates within six months of each other.

If a previously generated explanation held, its confidence would be augmented. If the confidence in the explanation, as determined by the decision function, $D_{useful-explan}$, was sufficient, a new search control rule would be generated. Then, this and other rules would be used to guide search in future scheduling runs. If, in this example, the confidence was high, the search control rule shown in Figure 2 would be created.

Evaluating the Method

In this section we describe the empirical analysis of the PEBL method.

Empirical Results

To evaluate the success of this approach we randomly generated sixteen sets of missions with varying proximity of launch dates and various carrier types. For the training phase, we used six sets of missions (with

TABLE 1: THE PLAUSIBLY-CHRONIC-RESOURCE-BOTTLENECK-ACROSS-TWO-MISSIONS GENERALIZATION PROBLEM

Given:

- *Target Concept:* Class of instances of a chronic resource bottleneck across two missions where:

  PLAUSIBLY-CHRONIC-RESOURCE-BOTTLENECK-ACROSS-TWO-MISSIONS(*task1.res*)
  ⟺
  IN-MISSION(*task1.miss1*) ∧ CONFLICTING-TASKS(*task1.res.task-list*)
  ∧ ISA(*miss2*.MISSION) ∧ NOT-EQUAL(*miss1.miss2*) ∧ NEARBY(*miss1.miss2*)
  ∧ CONFLICT-IN-MISSION(*task-list.miss2.task2*) ∧ USES-CARRIER(*miss1.carr1*)
  ∧ USES-CARRIER(*miss2.carr2*) ∧ CARRIER-TYPE(*carr1.carr-type1*)
  ∧ CARRIER-TYPE(*carr2.carr-type2*) ∧ RESOURCE-TYPE(*res.res-type*)
  ∧ TASK-TYPE(*task1.task-type1*) ∧ TASK-TYPE(*task2.task-type2*).

- *Training Example:*

  RESOURCE-BOTTLENECK(spacelab-systems-experiment-test1.
  automatic-test-equipment1)
  ISA(spacelab-systems-experiment-test1.SPACELAB-SYSTEMS-EXPERIMENT-TEST)
  IN-MISSION(spacelab-systems-experiment-test1.sts-62)
  ISA(sts-62.MISSION)
  USES-CARRIER(sts-62.lm2-1)
  ISA(lm2-1.LONG-MODULE-2)
  •••

- *Domain Theory:*

  USERS(*res.task-list*) ∧ EQ(*parallel-tasks*. {*task* ∈ *task-list* | INTERSECTS(*task.task1*)})
  → CONFLICTING-TASKS(*task1.res.parallel-tasks*)

  START-TIME(*task1.st1*) ∧ END-TIME(*task1.et1*) ∧ START-TIME(*task2.st2*)
  ∧ END-TIME(*task2.et2*) ∧ (LEQ(*st1.et2*) ∧ LEQ(*et2.et1*)) ∨(LEQ(*st1.st2*) ∧ LEQ(*st2.et1*))
  → INTERSECTS(*task1.task2*)

  ISA(*m1*.MISSION) ∧ ISA(*m2*.MISSION) ∧ NOT-EQUAL(*m1.m2*) ∧ LAUNCH(*m1.time1*)
  ∧ LAUNCH(*m2.time2*) ∧ MINUS(*time1.time2.diff*) ∧ ABSOLUTE-VALUE(*diff.abs-val*)
  ∧ LESS-THAN(*abs-val*.SIX-MONTHS) → NEARBY(*m1.m2*)

  CONFLICTING-TASKS(*conf-task.res.task-list*) ∧ ∃ *task* ∈ *task-list* IN-MISSION(*task.miss*)
  → CONFLICT-IN-MISSION(*task-list.miss.task*)

  ISA(*task*.SPACELAB-SYSTEMS-EXPERIMENT-TEST)
  → TASK-TYPE(*task*.SPACELAB-EXPERIMENT-INTEGRATION)

  ISA(*task*.PAD-OPERATIONS) → TASK-TYPE(*task*.LEVEL-I)
  •

  EQ(*res*.AUTOMATIC-TEST-EQUIPMENT1)
  → RESOURCE-TYPE(*res*.AUTOMATIC-TEST-EQUIPMENT)

  EQ(*res*.EAST-CRANE) → RESOURCE-TYPE(*res*.CRANE)
  •

  ISA(*c*.MPESS) → CARRIER-TYPE(*c*.MISSION-PECULIAR)

  ISA(*c*.LONG-MODULE-1) ∨ ISA(*c*.LONG-MODULE-2) ∨ ISA(*c*.PALLET-IGLOO)
  → CARRIER-TYPE(*c*.BIOLOGICAL)
  •••

- *Operationality Criterion:*

  The concept definition must be expressed in terms of the predicates used to describe the examples (e.g.. IN-MISSION(*task.miss*). USES-CARRIER(*miss.carr*)) or other selected. easily evaluated predicates from the domain theory (e.g..MINUS. LAUNCH. TASK-TYPE).

Determine:

A generalization of the training example that is a sufficient concept definition for the target concept and that satisfies the operationality criterion.

```
If
  TASK-TYPE(task1,
        SPACELAB-EXPERIMENT-INTEGRATION) ∧
  RESOURCE-TYPE(res,
             AUTOMATIC-TEST-EQUIPMENT) ∧
  IN-MISSION(task1,miss1) ∧
  LAUNCH(miss1,time1) ∧
  ISA(miss2,MISSION) ∧
  LAUNCH(miss2,time2) ∧
  MINUS(time1,time2,diff) ∧
  ABS-VAL(diff,abs-val) ∧
  LESS-THAN(abs-val,SIX-MONTHS) ∧
  USES-CARRIER(miss1,carr1) ∧
  CARRIER-TYPE(carr1,BIOLOGICAL) ∧
  USES-CARRIER(miss2,carr2) ∧
  CARRIER-TYPE(carr2,BIOLOGICAL) ∧
Then
  SCHEDULING-PERSPECTIVE(task1,
                             RESOURCE-BASED)
```

Figure 2: An Example Search Control Rule

$\sim$ 300 examples of resource contention). In our current implementation of PEBL, the decision function, $D_{useful-explan}$, is a simple threshold value. We analyzed the characteristics of the search space of scheduling this training set of missions to determine a likely range of threshold values. In this problem, a rule was created from an explanation if the explanation was applicable to four examples in a sample of one hundred. After training, we compared the respective speeds and WIP times for the resulting schedules generated with and without the learned search control rules created by PEBL, and for the schedules generated with rules created by using a zero threshold on ten different sets of missions. Using a threshold of zero is analogous to using standard EBL, where each explanation would be used to create a search control rule. This falsely implies that every resource bottleneck is chronic, however, so we refer to this as the "Learn Always" method. The results of the efficiency gain of synthesizing schedules using the learned knowledge are shown in Table 2. The results of the increased WIP times for the schedules generated with search control rules are shown in Table 3.

The number of tasks in each set of missions varied from 362 in Test-3 to 652 in Test-8. From 296 examples, PEBL created 19 search control rules. The number of times that the search control rules applied in each set of missions varied from 70 times in Test-3 to 138 times in Test-7 and Test-10. The rules were tested before scheduling each task. When using the Learn Always technique, 78 search control rules were created. The number of times the rules applied ranged from 117 times in Test-3 to 233 times in Test-10.

Discussion of the Results

In any learning system that generates search control rules, the utility of the rules is of primary importance [Minton, 1988b, Keller, 1987]. Minton defines search control knowledge as being *useful* when the cumulative benefits of applying the knowledge outweigh the cumulative costs of testing whether the knowledge is applicable. He defines performance improvement in terms of run time speed-up. But utility can also be measured in terms of solution quality. In many scheduling domains, the optimality of a schedule cannot be compromised for efficiency gain. Thus, when measuring the utility of the search control knowledge in scheduling, one must also take into account the quality of the generated schedule. In this domain, the optimization criterion for schedule quality is minimal WIP time.

In our experiments, the empirical component of PEBL proved essential. With too many rules, scheduling with the learned knowledge is worse than scheduling without the learned knowledge. The cost of testing the rules results in decreased system efficiency, and over-applying the non-optimizing search strategy creates poor schedules.

The results of our empirical analysis indicate that we must address the utility problem in terms of solution quality. The speed-up of the system is significant, ranging from $\sim$ 19 % to $\sim$ 77 % when the search control knowledge is applied. Test-9 showed the most significant efficiency improvement (76.99 %) because of the configuration of carriers and launch dates (three biological carriers with launch dates within six months, thus, high contention for human resources). The overall average speed-up was 34.25 %. However, the increase in WIP time varies from one set of missions to another. In some instances, the degradation in the quality of the synthesized schedules is noticeable, in others, the total WIP time is maintained. In this and other real-world domains, however, the ability to rapidly generate viable schedules is essential, and in some cases, worth marginal increases in WIP time.

When scheduling from the resource perspective, we choose resources randomly and then choose times based on these resources' availability. Some processing to select the "best" resource (the resource that has the maximum availability or that has a sufficient quantity available during a time interval closest to the desired time) may improve the synthesized schedules, because the choice of resource determines the values for start and end times when scheduling from the resource perspective. This is similar to the look-ahead schemes of [Smith and Ow, 1985] however, and would incur additional costs to the efficiency of the system.

Additionally, we believe that scheduling tasks from individual missions such that WIP time is minimized does not necessarily create schedules in which total WIP time is minimized. That is, there may be some cases when a sparser distribution of resource utilization in one mission allows the tasks in a second mission to

Sets of Test Missions	Cumulative Time (CPU sec)			% Improvement (with PEBL)	% Improvement (Learn Always
	Without Search Control	Search Control (with PEBL)	Search Control (Learn Always)		
Test-1	2189	1732	2973	20.88	-35.82
Test-2	1340	1066	2593	20.45	-93.51
Test-3	1292	817	1783	36.76	-38.00
Test-4	1476	1201	2785	18.63	-88.69
Test-5	2305	1469	2586	36.27	-12.19
Test-6	1951	1280	3044	34.39	-56.02
Test-7	2062	1533	2794	25.65	-35.50
Test-8	2853	1779	3381	37.64	-18.51
Test-9	4020	925	2287	76.99	43.11
Test-10	2662	1736	3801	34.79	-42.79

Table 2: Program Efficiency with and without the Learned Search Control Knowledge

Sets of Test Missions	Work-In-Process Time (days)			Increase in WIP (with PEBL)	Increase in WIP (Learn Always)
	Without Search Control	Search Control (with PEBL)	Search Control (Learn Always)		
Test-1	748.98	766.96	822.03	17.98	73.05
Test-2	787.97	911.00	926.55	123.03	138.58
Test-3	731.01	731.01	770.09	0	39.07
Test-4	1009.03	1009.03	1104.04	0	95.01
Test-5	550.49	617.04	719.04	66.55	168.55
Test-6	840.07	1021.08	1141.08	181.01	301.01
Test-7	916.02	1003.08	1052.07	87.06	136.05
Test-8	1021.04	1021.04	1113.08	0	92.04
Test-9	748.06	829.08	825.09	81.02	77.03
Test-10	997.04	1011.09	1148.06	14.05	151.02

Table 3: WIP time with and without the Learned Search Control Knowledge

better interleave with the tasks in the first mission, thus reducing total WIP time across both missions.

Related Work

The PEBL approach extends previous work in EBL [Mitchell *et al.*, 1986, Minton, 1988a, DeJong and Mooney, 1986] by applying it to a real-world problem and by adding an empirical component to confirm plausible explanations. In the real-world domain of payload processing scheduling, contention for a bottleneck resource is only plausibly chronic. This observation inspired the integration of plausibility into EBL, and ultimately, the creation of PEBL. The PEBL method is similar to other empirical learning techniques [Pazzani *et al.*, 1986, Danyluk, 1989, Hirsh, 1989, Mooney and Ourston, 1989]. However, our empirical component is not used to make inductive leaps. Rather, the target concept itself cannot be proved without a distribution of examples.

In the PEBL framework, the decision to create a rule is determined by the expected cost to solution quality, the conditional probabilities associated with plausible implications, and the expected match cost of applying a rule. Prior work [Minton, 1988b] in learning search control rules calculates the utility of a rule as a function of the frequency of application, the match cost and the efficiency gain. This is similar to work in learning macro-operators [Iba, 1989]; a macro-operator is deemed useful if its expanded length does not exceed some threshold, and if it appears in a successful solution path. Since our system does not create rules from every plausible explanation, we avoid having to reject useless or incorrect knowledge.

Finally, we extend previous work in scheduling using multiple perspectives [Smith and Ow, 1985] by applying machine learning techniques to minimize the changes in the search strategy to the most critical points in the search space.

Conclusions and Future Work

In the domain of scheduling, the balance between program efficiency and solution quality is essential. As our preliminary results are promising, we plan to thoroughly test these ideas on a set of comprehensive experiments. We will incorporate some look-ahead processing for resource selection and investigate other variable commitment search strategies. We will repeat the tests shown above, this time randomly varying the resource capacities. We intend to further analyze the characteristics of a good interleaving of tasks across missions. If we can incorporate these characteristics into the domain theory, we can then create search control rules that are likely to minimize WIP time. Because the number of generated search control rules effects the utility of the learned knowledge, accurately deciding whether a rule will be useful is essential. We intend to replace our current implementation of $D_{useful-explan}$ as a simple threshold, by adding heuristic measures to determine good solutions, and by extending the domain theory to include conditional probabilities that conjectured implications will be proven.

In the future, we plan to extend our work in machine learning and scheduling. We intend to apply PEBL to different search frameworks (e.g., constraint-based simulated annealing [Zweben, 1990]) and will investigate the idea of forming macro-constraints. Although individual constraints may not be restrictive, small subsets can effectively prune the search space. The application of these macro-constraints may quickly restrict the possible values for a variable. Other ideas include learning constraint ordering to find failure points early in the search process.

Acknowledgements

We thank Steve Minton for many valuable discussions and the use of Prodigy's rule system, John Bresina for his ideas about plausibility and John, Steve, Smadar Kedar, Pat Langley and Peter Friedland for carefully reviewing this paper.

References

[Brown, 1987] Brown, Richard H. Knowledge-based Scheduling and Resource Allocation in the CAMPS Architecture. In *Proceedings from the IEEE International Conference on Expert Systems and the Leading Edge in Planning and Control.* Benjamin/Cummings, 1987.

[Danyluk, 1989] Danyluk, A. Finding New Rules for Incomplete Theories. In *Proceedings of the Sixth International Workshop on Machine Learning*, 1989.

[DeJong and Mooney, 1986] DeJong, G.F., Mooney, R. Explanation-Based Generalization: An Alternative View. *Machine Learning*, 1, 1986.

[Fox, 1987] Fox, Mark S. *Constraint-Directed Search: A Case Study of Job Shop Scheduling.* Research Notes in AI, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

[Hankins *et al.*, 1985] Hankins, G.B., Jordan, J.W., Katz, J.L., Mulvehill, A.M., Dumoulin, J.N., Ragusa, J. EMPRESS: Expert Mission Planning and Re-planning Scheduling System. In *Expert Systems in Government Symposium*, 1985.

[Hirsh, 1989] Hirsh, H. Combining Empirical and Analytical Learning with Version Spaces. In *Proceedings of the Sixth International Workshop on Machine Learning*, 1989.

[Iba, 1989] Iba, G.A. A Heuristic Approach to the Discovery of Macro-operators. *Machine Learning*, 3(4), 1989.

[Keller, 1987] Keller, R.M. Defining Operationality for Explanation-Based Learning. In *AAAI-87 Proceedings*, 1987.

[Keller, 1987] Keller, R.M. Defining Operationality for Explanation-Based Learning. In *AAAI-87 Proceedings*, 1987.

[Laird *et al.*, 1986] Laird, J., Rosenbloom, P., Newell, A. *Universal Subgoaling and Chunking*. Kluwer Academic, Hingham, MA, 1986.

[Minton, 1988a] Minton, S. *Learning Effective Search Control Knowledge: An Explanation-based Approach*. PhD thesis, Carnegie Mellon University, 1988.

[Minton, 1988b] Minton, S. Quantitative Results Concerning the Utility of Explanation-Based Learning. In *AAAI-88 Proceedings*, 1988.

[Mitchell *et al.*, 1986] Mitchell, T.M., Keller, R.M., Kedar-Cabelli, S.T. Explanation-Based Learning: A Unifying View. *Machine Learning*, 1, 1986.

[Mooney and Ourston, 1989] Mooney, R., Ourston, D. Induction over the Unexplained: Integrated Learning of Concepts with Both Explainable and Conventional Aspects. In *Proceedings of the Sixth International Workshop on Machine Learning*, 1989.

[Pazzani *et al.*, 1986] Pazzani, M., Dyer, M., Flowers, M. The Role of Prior Causal Theories in Generalization. In *AAAI-86 Proceedings*, 1986.

[Pearl, 1985] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[Smith and Ow, 1985] Smith, S.F., Ow, P.S. The Use of Multiple Problem Decompositions in Time-Constrained Planning Tasks. In *IJCAI-85 Proceedings*, 1985.

[Zweben and Eskey, 1989] Zweben, M., Eskey, M. Constraint Satisfaction with Delayed Evaluation. In *IJCAI-89 Proceedings*, 1989.

[Zweben, 1990] Zweben, M. A Framework for Iterative Improvement Search Algorithms Suited for Constraint Satisfaction Problems. Technical Report, NASA Ames Research Center, 1990.