Appears in the Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), pp. 721–726

Learning to Predict User Operations for Adaptive Scheduling

Melinda T. Gervasio and **Wayne Iba** and **Pat Langley**
Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, California 94306
{gervasio,iba,langley}@isle.org

Abstract

Mixed-initiative systems present the challenge of finding an effective level of interaction between humans and computers. Machine learning presents a promising approach to this problem in the form of systems that automatically adapt their behavior to accommodate different users. In this paper, we present an empirical study of learning user models in an adaptive assistant for crisis scheduling. We describe the problem domain and the scheduling assistant, then present an initial formulation of the adaptive assistant's learning task and the results of a baseline study. After this, we report the results of three subsequent experiments that investigate the effects of problem reformulation and representation augmentation. The results suggest that problem reformulation leads to significantly better accuracy without sacrificing the usefulness of the learned behavior. The studies also raise several interesting issues in adaptive assistance for scheduling.

Introduction

In recent years, there has been a surging interest in the mixed-initiative paradigm where multiple agents—specifically humans and software systems—share control by partitioning the responsibilities in problem solving. This trend holds not only for the ubiquitous applications on the Web, but also in domains such as planning and scheduling where the traditional AI approach has involved autonomous systems. Ideally, a mixed-initiative system benefits from the individual strengths of its constituents—the expertise of the human user and the computational power of the computer. For this synergy to occur, however, the division of responsibilities must be mutually beneficial and the two participants must be able to work together effectively. Meeting these requirements with a single, static system will be difficult because different users will have different abilities and preferences. In addition, as software systems become more powerful, they may outgrow the user's ability to communicate tasks or requests that best take advantage of those abilities. Machine learning can help address these problems by providing adaptive systems that automatically tailor their behavior to different users.

In this paper, we present an empirical study of learning user models in an adaptive scheduling assistant. We begin by describing our application, a synthetic domain that involves responding to hazardous materials incidents. We also describe INCA, the interactive assistant we are developing for this domain. Traces of user interactions with INCA provided the data for our learning experiments. We formulate the scheduling assistant's task of predicting user operations as a classification problem, and we discuss the results of a baseline study, which showed some benefit from learning but also left room for improvement. The three subsequent experiments investigate the effects of problem representation and formulation on performance. Their results show that problem reformulation can lead to much better adaptation without sacrificing the usefulness of the learned concepts. The experiments raised a number of issues, which we consider in our closing discussion of related and future work.

Scheduling for Crisis Response

A dominant theme in crisis response is *urgency*—an agent is compelled to act to avert an undesirable situation in a limited amount of time. Finding an efficient level of interaction between the human user and the computer is thus particularly important. Our response to urgency relies on machine learning to acquire user models to facilitate this interaction. To illustrate our ideas and to lay the groundwork for the experiments in this paper, we will discuss them in the context of a synthetic hazardous materials domain (HAZMAT) and the INteractive Crisis Assistant (INCA) that we developed for this domain. INCA provides assistance for both planning and scheduling, but we will focus on the scheduling task here.

HAZMAT Response Using INCA

In developing the synthetic HAZMAT domain, we consulted the 1996 North American Emergency Response Guidebook (Transport Canada et al., 1996), a handbook for first responders to hazardous materials inci-

dents. A HAZMAT problem consists of a spill and possibly a fire involving one of 50 types of hazardous material. There are 4000 classes of HAZMAT incidents, varying in the type and amount of material involved, the location of the incident, and the characteristics of the spill and any fire. Incidents also have associated fire and health hazards that, respectively, characterize the probability of a fire and the danger to people's health.

There are 49 types of actions and 25 types of resources available for responding to a HAZMAT incident. In any given problem, only a subset of the actions will be applicable, as indicated in the Guidebook. Each action addresses particular aspects of a HAZMAT problem and requires some minimum set of resources. The specific resources available, and their associated capacity and quantity constraints, vary with every problem.

INCA is an interactive system that provides planning and scheduling assistance for HAZMAT response. In the planning phase, the user interacts with INCA to choose the *schedulable* actions, a subset of the applicable actions to the problem. The input to the scheduling phase is this set of schedulable actions, the set of available resources, and an initial (possibly empty) candidate schedule provided by INCA.

In a departure from traditional scheduling, the task is to choose some *subset* of the schedulable actions and to assign them to resources. Moreover, actions may be allocated a variable number of resources and arbitrary duration. Consider the action of extinguishing a fire with a hose. Allocating more resources (firefighters, hoses, hydrants, etc.) to the action, as well as simultaneously scheduling other extinguishment actions (e.g., extinguish with dry sand), will put out the fire more quickly. This interdependence among actions, resources, and effects makes it difficult to completely determine resource requirements and action durations prior to scheduling. With effective heuristics being difficult to engineer for the resulting underconstrained problem, machine learning becomes even more attractive.

Scheduling an action involves four decisions: the number of resources to allocate to the action, the specific resources to allocate, the start time, and the duration. Scheduling in INCA takes place in a repair space, where the operators include adding and removing actions from the schedule as well as modifying parameters of scheduled actions. Specifically, the user interacts with INCA through a graphical interface that provides the user with five scheduling operators: add a new action, remove an action, shift the start time of an action, change the duration of an action, and switch an action from one resource to another.

The final schedule must be *feasible*—that is, it must not violate any capacity or quantity constraints. A resource is *oversubscribed* if there is any time at which the number of simultaneously scheduled actions on that resource exceeds its capacity. Similarly, a resource is *overallocated* if the actions scheduled on it consume more than the available resource quantity. An *infeasibly scheduled* action is one that participates in the oversubscription or overallocation of any resource. Using the five available operators, the user modifies the schedule until it is feasible and he considers it acceptable.

INCA currently assists the user in scheduling by providing an initial candidate schedule retrieved from a case library, by suggesting heuristically determined default values for resources, durations and start times, and by checking the feasibility of schedules. By integrating learning into INCA, we hope to improve its assistant capabilities by letting it adapt its behavior to individual users.

Acquiring and Applying User Models

A user will have personal beliefs about what actions are appropriate for a problem, what actions are more important than others, and what actions should be performed first. These preferences are reflected by the operators that the user selects and their results—what actions are included and how they are ordered in the schedule, how resources are allocated to different actions, and how different conflicts are resolved. The goal of learning in INCA is to extract such information from traces of its interactions with the user and to use this information to adapt its behavior accordingly. For example, INCA might use this preference information to suggest resources, durations, and start times that are similar to the user's previous choices when adding the action. Or it might apply scheduling operators based on this information to specifically tailor initial candidate schedules. By making suggestions that the user is more likely to accept, INCA can make the HAZMAT response process more efficient, thereby directly addressing the urgency aspect of crisis response.

As our first step in integrating learning into INCA, we focused on the prediction of scheduling operations. Specifically, INCA's user modeling task was: Given a particular scheduling state—as characterized by the problem, available resources, schedulable actions, and current schedule—predict the user's next scheduling operation. This can be translated into a standard classification task, with the class being the scheduling operation and the instance being the scheduling state for which the prediction is being made.

Baseline Study: Effects of Learning

We extracted the data for our learning experiments from the individual traces of two users, each interacting with INCA over 140 HAZMAT problems. Each scheduling operation performed while solving a problem corresponds to a training or test instance. From traces of the first and second users, we extracted 935 examples (data set A) and 1049 examples (data set B) respectively.

Nearly all fielded applications of machine learning (Langley & Simon, 1995) rely on attribute-value representations and standard supervised induction methods, so we decided to investigate the feasibility of such an approach here. This has the advantage of either avoiding the cost of engineering a special-purpose approach

Table 1: Percentage accuracy in predicting user operations during scheduling.

	A	B
test on same user	22.15 ± 2.22	26.87 ± 2.03
test on other user	18.70 ± 0.58	18.67 ± 0.58

or justifying the need for more complex representations and algorithms.

For the baseline study, we used 86 attributes to describe the scheduling state. We represented the problem using 12 nominal attributes directly corresponding to the material, spill, fire, and hazard features of the HAZMAT problem. We represented the resources with 25 Boolean features, one for each resource type, with the value denoting whether there is at least one resource of that type available. Finally, we represented the schedulable set of actions and the schedule with 49 attributes corresponding to the 49 possible actions. These attributes had four possible values: not schedulable, unscheduled, feasibly scheduled, or infeasibly scheduled.

There are several levels at which the prediction task can be formulated. At the highest level is the prediction of a general scheduling operator such as ADD *any* action. At the lowest level is the prediction of a specific instantiation of a scheduling operator such as ADD the `absorb-with-dry-sand` action on `crew members #1` and `#4` and `dry sand #2` starting at time `21` for a duration of `10` time units. For the baseline study, we chose an intermediate level, requiring the specification of a particular action but not specific resources or amounts. We also combined the remove action, change duration, shift action, and switch resource operators into a single REPAIR operator. This resulted in ADD and REPAIR operations for each of the 49 actions, for a total of 98 different classes.

We can thus state the learning task as: Given a set of training examples, learn a classifier that makes correct predictions on new examples. Preliminary studies with supervised learning algorithms from the repository maintained by the Machine Learning Group at the University of Texas at Austin revealed ID3 (Quinlan, 1986) to be the most promising, so we chose to use this in our formal experiments.[1]

We had two hypotheses for the baseline experiment. The first was that learning would improve accuracy, and the second was that greater gains would result if we trained and tested on data from the same user than if we trained on data from one user and tested on data from another. The aim of learning is to model a specific user, so we expected some benefit from learning on another user but not as much as learning from the same user.

We tested these hypotheses by first running ten trials on each user's data set, with each trial using 800 randomly chosen examples for training and the rest for testing. We then ran another ten trials, this time using the entire other user's data set for testing. Table 1 shows the results for each data set, averaged over the ten runs for each condition (95% confidence intervals). The results suppport both our hypotheses: under each condition, the resulting accuracies were notably better than guessing randomly (1.02%) and guessing the most frequent class (8.57%), with the performance from training and testing on data from the same user being significantly better than that from training on one user and testing on another (paired t test, $p < 0.01$).

[1] The repository resides at www.cs.utexas.edu/users/ml. We obtained similar results with the C4.5 system.

Table 2: Percentage accuracy in predicting user operations after various problem reformulations.

	A	B
temporal attributes	18.81 ± 1.73	26.18 ± 1.68
abstracted	42.30 ± 3.15	36.63 ± 2.06
alternative classes	59.33 ± 2.39	64.30 ± 2.53
no illegal predictions	78.24 ± 1.66	76.61 ± 1.73

Improving Accuracy Through Problem Reformulation

The baseline study showed that learning improved performance but, even in the best case (26.87% for data set B, within-user test), there were more than twice as many misclassifications as correct classifications. This calls into question the utility of the predictions in the context of a scheduling assistant. Our analysis of the results suggested our formulation of the problem as a prime suspect for this poor performance. We identified three ways to reformulate the problem: adding contextual temporal information, abstracting the class labels, and modifying the prediction task. We now describe the three experimental studies we conducted to test whether these problem reformulations would increase predictive accuracy. We also propose a modification to the prediction element that promises substantial performance improvement.

Adding Temporal Information

In the first experiment, we wanted to determine the effects of augmenting the problem representation with information about the other operators used in solving the problem. Our statement of the performance task requires INCA to predict the user's *immediate* next operation. This is likely to depend on the other operators the user selects within the same problem, particularly the most recent operators selected, but the base problem representation included no such explicit contextual information.

Our hypothesis in this experiment was that adding contextual temporal information would improve accuracy. To test this hypothesis, we extended the instance representation with five attributes to represent the five most recent operations the user performed prior to the

Table 3: Total percentage of prediction errors and percentage due to illegal predictions.

	A		B	
	total	illegal	total	illegal
base condition	77.85	19.78	73.13	14.94
temporal attributes	81.18	41.11	73.82	36.95

operation associated with that instance.[2] We then ran ten trials on each data set using the same training and test splits as in the baseline study.

Table 2 and Figure 1 present the results (95% confidence intervals) from the problem reformulation experiments. The results from the first experiment (Table 2, temporal attributes) did not support our hypothesis: temporal information did not affect accuracy. One explanation lies in the number of errors that involved predicting an illegal operation. Incorrect predictions may be divided into two types: *legal* predictions that correspond to operations that can be performed in the scheduling state and *illegal* operations that cannot. Specifically, an ADD-*action* operation is illegal if the *action* is already in the schedule or it is not in the schedulable set; a REPAIR-*action* operation is illegal if the *action* is not in the schedule. Table 3 shows that the percentage of examples that were misclassified into illegal operations increased with the introduction of the temporal attributes to account for over half of all the misclassifications. These results reveal that the problem representation is not sufficiently balancing the necessary contextual and legality information to let the system make good, legal predictions. They also suggest additional analysis and experimentation on problem representation to shed light on this issue.

Class Abstraction

In the second experiment, we wanted to investigate the utility of simplifying the problem via class abstraction. As stated earlier, the prediction task can be formulated at different levels, and for the baseline study we chose an intermediate level requiring the specification of an ADD or REPAIR on a particular action but not the specification of particular resources or amounts.

Our hypothesis in the second experiment was that performance would improve on a more abstract prediction task. To test this hypothesis, we chose to abstract one level to require predicting an ADD/REPAIR operation on a class of actions rather than on individual actions. For example, the abstract ADD "extinguish with hose" replaces the set of specific ADDs on actions such as "extinguish with hose using water from a hydrant" and "extinguish with hose using foam and a pumper". We felt this was a reasonable abstraction in that it makes few additional demands on the user,

[2]We tried adding one to five previous operations and found no significant differences.

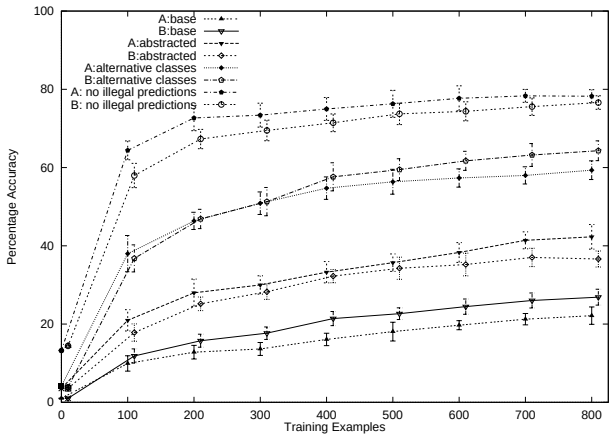

Figure 1: Learning curves for the various experimental conditions, showing improvement due to successive problem reformulations.

who must only choose from a small set of corresponding more specific actions. Abstracting the problem in this manner reduced the original 98 classes into 28 abstract classes, and also increased the proportion of the most frequent class from 8.57% to 11.19%. We ran ten trials on each data set, using the same training and test splits as in the baseline study.

The results support our hypothesis: accuracy increased significantly (paired t test, $p < 0.001$) for both data sets with class abstraction (Table 2 and Figure 1, abstracted). This result may not be surprising since the abstraction results in a simpler prediction task. However, it raises the issue of determining an appropriate task level in mixed-initiative settings such as interactive scheduling. Formulating the prediction task at higher levels transfers more decision-making responsibility to the human user, but this is not undesirable if it leads to more effective interaction overall.

Redefining Correctness

In the third experiment, we wanted to investigate the effects of modifying the prediction task by allowing alternative class labels. The formulation of the learning problem as the problem of predicting the user's next operation was a natural fit to the data provided by the user traces. However, it is an unnecessarily difficult learning task in that it requires the system to predict a user's immediate next action. A user may arrive at the same schedule through different sequences of operations, many of which may be equally acceptable to the user. Thus, a more appropriate learning task might be to predict any one of the user's subsequent operations, subject to legality given the current state.

Our hypothesis in this experiment was that accuracy would improve under the redefined correctness criterion. To test this hypothesis, we first modified the original examples to include a set of alternative classes, consisting of all the subsequent operations the user in-

voked to solve the HAZMAT incident minus those that were illegal in the current state. We ran ten trials on each data set, again using the same training and test splits. Training proceeded as before but, during testing, we judged a prediction to be correct if it matched the instance's label or one of its alternative classes.

The results support our hypothesis: significantly greater accuracy (paired t test, $p < 0.001$) was achieved on the reformulated task (Table 2 and Figure 1, alternative classes). Again, the results might seem obvious since allowing alternative classes simplifies the prediction task. However, as stated earlier, this may in fact be a more appropriate task in that it may better capture what concerns users in the scheduling process.

Correcting for Illegal Operations

The observation that many errors involved illegal operations (Table 3) suggests another change to the prediction element. The revised system would check to determine if a given prediction is illegal and, if so, would move on to the next most likely prediction, continuing in this manner until it predicts a legal operation. Implementing this scheme is straightforward for some classifiers, like naive Bayes and nearest neighbor, which rank their predictions, but it is more complicated for decision trees.

We are still in the process of considering various approaches to implementing this filtering scheme. However, we can estimate the new accuracy by factoring out the existing illegal predictions. Let us assume that the learned predictor's accuracy will be the same on the illegal classifications as on the legal ones. Let a be the accuracy with abstraction, s be the gain from alternative classes, and i be the misclassifications that were illegal. The accuracy under this new scheme would be $a + s + \frac{a+s}{100} i$. This results in accuracies in the 75 to 80% range (Table 2 and Figure 1, no illegal predictions), which are much more promising than the baseline results.[3]

Related and Future Work

Previous work on learning for scheduling has focused on autonomous systems, with the aim of improving scheduling efficiency or schedule quality. For example, Gratch and Chien (1993) increased efficiency in a deep space network scheduler by acquiring effective domain-specific heuristics. Similarly, Eskey and Zweben (1990) increased scheduling efficiency in payload processing for the NASA space shuttle by acquiring rules to avoid chronic resource contention. Zhang and Dietterich (1995) used reinforcement learning on the same task, but to acquire heuristics that resulted in shorter schedules. Our framework for learning differs in its emphasis on acquiring user-specific performance criteria and, as a result, in its reliance on detailed traces of user decisions.

[3]For naive Bayes, we have obtained actual results that were as good or better than these estimates.

Some recent AI scheduling systems (e.g., Smith et al., 1996; Fukunaga et al., 1997) have mixed-initiative modes, but few incorporate learning or adaptation to different users. The same holds for AI crisis response systems such as O-Plan2 (Tate et al., 1994) and SOCAP (Bienkowski, 1996). An exception is CABINS (Miyashita & Sycara, 1995), an assistant for job-shop scheduling that learns user preferences on repair heuristics. Like INCA, CABINS acquires preferences from user traces and uses these to direct a repair-space search for a solution. CABINS differs in that it invokes a heuristic scheduler to generate an initial schedule and a case-based method to learn user preferences, whereas INCA uses a case-based method to retrieve an initial schedule and other learning techniques to acquire user preferences. We believe this approach provides a better fit to crisis domains, where case libraries can be built from standard operating procedures or the results of planning exercises.

Much work on personalization, particularly for computer-aided instruction, has focused on methods for constructing models of user behaviors that can then be used to classify users and modify system interaction accordingly (e.g., Sleeman & Smith, 1981; Clancey, 1979; Anderson & Reiser, 1985). Langley (1997) uses the term *adaptive user interfaces* to refer to systems like INCA that instead use machine learning to construct user models from interaction traces. Previous work in this area include Schlimmer and Hermens' (1993) interface for repetitive form filling; Dent et al.'s (1992) CAP, a calendar apprentice for scheduling appointments; and Pazzani et al.'s (1996) SYSKILL & WEBERT, a Web page recommendation service. Like INCA, these systems use their learned user models to tailor their behavior to individual users. The work described in this paper differs in its focus on an explicit evaluation of the factors affecting success.

A priority for future work involves fully integrating the learned predictors into INCA, which would let us directly evaluate our hypothesis that learning user preferences will produce more rapid HAZMAT response. This will also let us test our hypotheses about appropriate task formulations. Initially, we plan to use an offline setting, in which we train INCA on user traces, incorporate the learned model into the system, and then let the same user interact with INCA while measuring performance. Eventually, we plan to integrate learning in an online setting, where the system updates its user model during the course of its interactions. In such settings, it becomes particularly important that the system avoid making suggestions that are so unacceptable and distracting that the user would have been better off without assistance. One response, which we plan to explore in future work, is to incorporate confidence measures into system predictions to prevent it from making suggestions until they exceed some minimum threshold.

Another important area for future work is task reformulation. The results on abstraction suggest that we look for simple learning problems that would ben-

efit from standard induction algorithms. One approach would be to divide the prediction task into a set of smaller tasks—for example, predicting specific resources, durations, and start times once an action has already been selected for scheduling; or proceeding with prediction hierarchically through more and more specific operations. The results on alternative classes suggest that we reevaluate our task definition in light of the kinds of suggestions (i.e., system predictions) the user will find helpful. We are currently exploring alternative formulations of the learning task along these lines. We also expect to see greater benefits from combinations of successful reformulations.

We also need to revisit our representational options. The results on adding temporal attributes indicate that there may be important information that is currently not being captured. The attribute-value scheme is simple but it does not let us easily represent specific schedules (i.e., what actions use what resources over what time intervals) or the relationship between what actions address what problem features. This information, which may play a role in user preferences, can be more easily captured in a relational representation, which we plan to investigate in future versions of INCA.

Concluding Remarks

In this paper, we reviewed INCA, an adaptive assistant for crisis scheduling, and presented empirical studies of learning user models aimed at improving the system's behavior. We formulated the task of the scheduling assistant as predicting the user's action in response to a particular problem, set of resources, and current schedule. We set out to explore the limits of simple problem representations and established learning algorithms, and our initial study revealed some benefits from learning, but also left room for improvement.

Subsequent experiments demonstrated that abstractions of the learning task, as well as reformulations of the prediction task, result in substantially better performance without sacrificing, and possibly even improving, the usefulness of the learned behavior. These findings suggest that adaptive interfaces like INCA, which learn models of user preferences, constitute a promising approach to mixed-initiative scheduling that deserves fuller exploration in future research.

Acknowledgments. This research was supported by the Office of Naval Research under Grant N000014-96-1-1221. We would also like to thank Mark Maloof for his helpful comments on the paper, and the Organizational Dynamics Center group at Stanford University for many interesting discussions on crisis response.

References

Anderson, J. R. and Reiser, B. J. 1985. The LISP Tutor. *Byte* 10:159–175.

Bienkowski, M. 1996. SOCAP: System for Operations Crisis Action Planning. In *Advanced Planning Technology*, Tate, A., ed., 70–76. Menlo Park: AAAI Press.

Clancey, W. J. 1979. Dialogue Management for Rule-Based Tutorials. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*.

Dent, L.; Boticario, J.; McDermott, J.; Mitchell, T.; and Zaborowski, D. 1992. A Personal Learning Apprentice. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 96–103.

Eskey, M., and Zweben, M. 1990. Learning Search Control for Constraint-Based Scheduling In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 908–915.

Fukunaga, A.; Rabideau, G.; Chien, S.; and Yan, D. 1997 Toward an Application Framework for Automated Planning and Scheduling. In *Proceedings of the 1997 International Symposium on Artificial Intelligence, Robotics and Automation for Space*. Tokyo, Japan.

Gratch, J., and Chien, S. 1993. Learning Effective Control Strategies for Deep-Space Network Scheduling. In *Proceedings of the Tenth International Conference on Machine Learning*, 135–142.

Langley, P., and Simon, H. A. 1995. Applications of Machine Learning and Rule Induction. *Communications of the ACM* 38:55–64.

Langley, P. 1997. Machine Learning for Adaptive User Interfaces. In *Proceedings of the Twenty-First German Annual Conference on Artificial Intelligence*, 53–62.

Miyashita, K. and Sycara, K. 1995. CABINS: A Framework of Knowledge Acquisition and Iterative Revision for Schedule Improvement and Reactive Repair. *Artificial Intelligence* 76: 337–426.

Pazzani, M.; Muramatsu, J.; and Billsus, D. 1996. SYSKILL & WEBERT: Identifying Interesting Web Sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 54–61.

Quinlan, R. 1986. Induction of Decision Trees. *Machine Learning* 1:81–106.

Sleeman, D. H. and Smith, M. J. 1981. Modelling Pupil's Problem Solving. *Artificial Intelligence* 16:171–187.

Smith, S.; Lassila, O.; and Becker, M. 1996. Configurable, Mixed-Initiative Systems for Planning and Scheduling. In *Advanced Planning Technology*, Tate, A., ed., 235–241. Menlo Park: AAAI Press.

Tate, A., Drabble, B., and Kirby, R. B. 1994. O-Plan2: an Open Architecture for Command Planning and Control. In *Intelligent Scheduling*, Zweben, M. and Fox, M. S., eds. San Francisco: Morgan Kaufmann.

Transport Canada, the U.S. Department of Transportation, and the Secretariat of Communications and Transportation of Mexico. *1996 North American Emergency Response Guidebook*.

Zhang, W. and Dietterich, T. 1995. A Reinforcement Learning Approach to Job-shop Scheduling. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence*.