Scheduling with Uncertain Resources: Learning to Ask the Right Questions

Alexander Carpentier
Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
alexcarp@cs.cmu.edu

Mehrbod Sharifi
Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
mehrbod@cs.cmu.edu

Eugene Fink
Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
e.fink@cs.cmu.edu

Jaime G. Carbonell
Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
jgc@cs.cmu.edu

***Abstract*—We consider the task of scheduling a conference based on incomplete information about resources and constraints, which requires elicitation of additional data, and describe a learning procedure that improves elicitation strategies. We outline the representation of incomplete knowledge, and then describe an adaptive elicitation procedure, which learns to identify critical missing data.**

***Index Terms*—Uncertainty, optimization, elicitation, active learning.**

I. Introduction

When we work on a practical scheduling task, we may not have complete knowledge of the available resources and scheduling constraints. For example, when scheduling a conference, we may not know the exact attendance of specific events or preferences of specific speakers, which may complicate the selection of appropriate rooms. The task of scheduling under uncertainty involves several problems, including the representation of uncertain data, automated construction of schedules based on these data, and identification of critical missing knowledge.

We have developed a system for scheduling based on incomplete and partially uncertain data [1], which has been part of the RADAR project (www.radar.cs.cmu.edu) at Carnegie Mellon University, aimed at building a software agent for assisting an office manager. We have described this scheduling system in a series of four papers; specifically, we have explained the representation of uncertainty [2], search for near-optimal schedules [11], elicitation of additional data [3], and collaboration between the system and its user [10]. We now present a learning mechanism for improving elicitation strategies.

The manuscript was received on March 16, 2008. The described work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185.

The problem of eliciting additional data arises in a variety of applications, and researchers investigated several related techniques [9]. For example, Burke *et al.* [7][8] built an assisted browsing system that helped the user to construct queries, and Linden *et al.* [13], Pu and Faltings [14], and Torrens *et al.* [16] developed software agents that helped customers to find airline tickets by eliciting their flight preferences. The authors of these systems assumed that every query had only a few parameters, and their approach would be impractical for scheduling problems, which may have thousands of uncertain values. As another example, Boutilier *et al.* studied the optimization under uncertainty, and developed a procedure that generated questions about missing data [4][5][6][15]; however, it allowed uncertainty only in discrete parameters, and thus it would be inapplicable to scheduling with continuous variables.

We have considered the problem of scheduling a conference under uncertainty, which may require handling of thousands of uncertain values, including both discrete and continuous parameters. The examination of the previous elicitation techniques has shown that they are inapplicable to this problem, and we have developed a novel elicitation mechanism, based on evaluating the impact of missing data on the schedule quality [3]. We now present a learning procedure that improves the performance of this mechanism and helps to adapt it to new scheduling scenarios. We outline the representation of uncertain data (Section II), explain the elicitation steps (Section III), describe the related learning mechanism (Section IV), and give experiments on its effectiveness (Section V).

II. Scheduling

We begin by reviewing the representation of resources and constraints in the developed system. A more detailed description is available in the earlier paper on the representation of uncertainty in conference scheduling [2]. The two main objects in this representation are *rooms,* which define available resources, and *conference events,* which include related constraints.

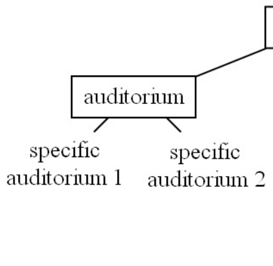

Fig. 1. Example of a room hierarchy.

Uncertain value:
prob$_1$: from *min*$_1$ to *max*$_1$
...
prob$_m$: from *min*$_m$ to *max*$_m$
where
$min_1 \le max_1 \le ... \le min_m \le max_m$
$prob_1 + ... + prob_m = 1$

Fig. 2. Representation of an uncertain value, which includes multiple disjoint intervals and their respective probabilities.

We describe a *room* by a set of properties, such as size, availability of overhead projectors and microphones, and distances to other rooms. We also specify the availability of each room, represented by a collection of time intervals. In addition, we may arrange rooms into a tree-structured hierarchy, which specifies main room types, as illustrated in Figure 1. The system allows manual specification of a room hierarchy, and it also includes a clustering procedure for automated construction and refinement of a hierarchy based on similarities among rooms.

The description of a *conference event* includes its importance and a set of related constraints and preferences. The importance is a positive integer, the constraints are ranges of acceptable values for start time, end time, duration, and room properties, and the preferences are ranges of preferred values, which must be within the respective acceptable ranges. The system allows arranging events into a tree-structured hierarchy, similar to the room hierarchy.

We may have incomplete information about room properties, event importances, constraints, and preferences. We represent every uncertain value by a probability density function, approximated by a set of uniform distributions. That is, we encode an uncertain value by a set of disjoint intervals that may contain it, with a probability assigned to each interval (Figure 2).

When constructing a schedule, the system assigns a specific room, start time, and duration to every event. We measure the quality of these assignments on the scale from 0.0 (lowest) to

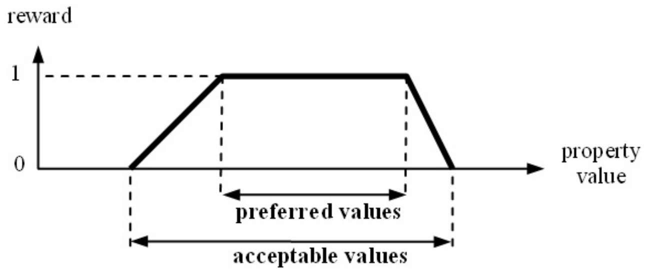

Fig. 3. Reward for satisfying a preference.

1.0 (highest). If an assignment violates any hard constraints, its quality is zero. Else, we compute its quality through the rewards for satisfying the related preferences. If a start time, end time, duration, or room property is within the preferred range, the respective reward is 1.0. If it is outside the preferred range, the reward linearly decreases with the distance from the range (Figure 3). If the event has k preferences, and their respective rewards are $r_1, ..., r_k$, then the assignment quality is $(r_1 + ... + r_k)/k$.

The overall schedule quality is the weighted sum of the quality values of individual assignments. That is, if a schedule includes n events, their quality values are $Qual_1, ..., Qual_n$, and their importances are $imp_1, ..., imp_n$, the overall quality is

$$\frac{imp_1 \cdot Qual_1 + ... + imp_n \cdot Qual_n}{imp_1 + ... + imp_n}.$$

If some relevant values are uncertain, the system evaluates the expected quality of each assignment, and then computes the expected value of their weighted sum:

$$\frac{E(imp_1) \cdot E(Qual_1) + ... + E(imp_n) \cdot E(Qual_n)}{E(imp_1) + ... + E(imp_n)}.$$

The purpose of scheduling is to maximize this expected quality. The system includes a scheduling procedure based on randomized hill-climbing, which does not guarantee optimality, but usually finds near-optimal solutions [11].

III. Elicitation

The system computes not only the expected schedule quality, but also its standard deviation. A high deviation means that the true quality may be significantly different from its expectation. We may reduce uncertainty by asking the human administrator to provide more accurate data; we list the types of possible questions in Figure 4.

The system includes two modules for ranking potential questions by importance, called *impact-based elicitor* and *adaptive elicitor* (Figure 5). After ranking all questions, it requests the user to provide answers, starting with the most important questions. If the user answers some questions, the system uses the new data to improve the schedule.

- Provide the exact value for an uncertain room property.
- Provide the exact value for an uncertain event importance.
- Provide the exact acceptable range for start time, end time, duration, or room property in an event description.
- Provide the exact preferred range for start time, end time, duration, or room property in an event description.

Fig. 4. Types of questions to the user. The system may ask to find out more information about room properties, event importances, and scheduling constraints and preferences.

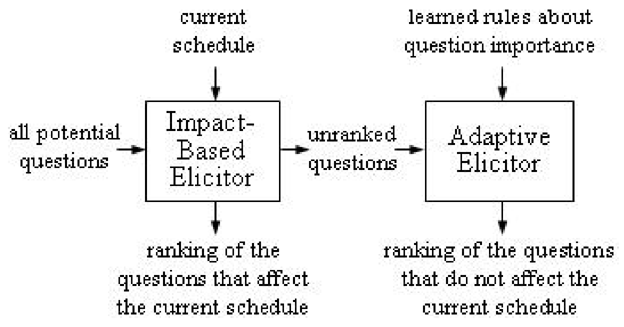
current schedule
learned rules about question importance
all potential questions
Impact-Based Elicitor
unranked questions
Adaptive Elicitor
ranking of the questions that affect the current schedule
ranking of the questions that do not affect the current schedule

Fig. 5. Two modules for the ranking of potential questions.

Impact-based elicitor: The first module is a combination of the two algorithms developed by Bardak [1][3], which evaluate the impact of potential questions on the quality of a given schedule. For each question, the system estimates the expected quality increase if it can use the related answer in re-scheduling. After evaluating each question, it sorts the questions by their impacts. The related algorithms ignore the questions that have no impact on the given schedule, even if the related answers may help in the future.

Adaptive elicitor: The second module uses heuristic rules to rank the questions ignored by the impact-based elicitor. Although the answers to these questions do not affect a given schedule, they may be essential for other schedules. For example, we may need to ask about the properties of a major auditorium even if the current schedule does not use it. The effectiveness of this module depends on the available heuristics, and we have developed a learning mechanism for improving them.

IV. Learning

The system learns typical importance values of potential questions, and then uses them to estimate the relative importance of questions that do not affect a given schedule. We explain the use of learned values and then present the learning procedure.

Question importances: The system keeps a list of all potential questions, which correspond to parameters of scheduling problems, including properties of specific rooms, event importances, and constraints and preferences of specific events. If the system encounters several scheduling scenarios, with different rooms and events, it adds all potential questions from all scenarios to the overall list. For each question, it keeps the learned value of its typical importance, along with a counter of all scheduling problems used in learning this value. When the adaptive elicitor processes the questions that do not affect the current schedule, it sorts them in the decreasing order of the learned importances.

Learning procedure: The system learns typical importances by applying the impact-based elicitor to a set of training problems. We summarize the learning algorithm in Figure 6; for each problem in the training set, it invokes the impact-based elicitor to determine the importances of the questions that affect the related schedule, and uses the results to update the learned importances. For each question, the learned importance is the mean of its importances in specific problems, and the respective value of the problem counter is the number of training problems where this question affects the schedule.

Generalization: If some questions do not affect any schedules in the training set, the procedure in Figure 6 does not learn their importances. We use the room and event hierarchies to estimate the importances of these questions based on the importances of other similar questions. We give the estimation procedure for questions about rooms in Figure 7; the procedure for questions about events is similar. These two procedures input a question without a learned importance, retrieve analogous questions for other objects of the same type, and estimate the importance of the given question as the mean importance of the retrieved questions.

Problem generation: If the available set of training problems is too small, the system generates new variations of given problems, and uses them as additional training instances. To construct a new problem, it randomly changes event assignments in a given schedule and then applies the hill-climbing procedure to repair the schedule. Since different starting points of the hill-climbing usually lead to different local maxima, multiple applications of this procedure give different near-optimal schedules, which serve as different training problems.

V. Experiments

We have applied the developed system to schedule a four-day conference, which includes eighty-four events, where each event has twenty constraints. The available resources include eighty-eight rooms, where each room has seventeen properties. The problem representation comprises about 3300 parameters;

```
Input: Set of scheduling problems, denoted problem-set.
Output: Set of questions with learned importances, learned-qs.

LEARN(problem-set)
initialize learned-qs as an empty set of questions
for each problem in problem-set do
    apply the impact-based elicitor to this problem,
        which returns the set of selected questions, selected-qs,
        and the impact of each selected question, current-impact
    for each question $q$ in selected-qs do
        if $q$ is not in learned-qs
        then add $q$ to learned-qs
                total-impact$[q]$ = current-impact$[q]$
                counter$[q] = 1$
        else total-impact$[q]$ = total-impact$[q]$ + current-impact$[q]$
                counter$[q]$ = counter$[q] + 1$
for each question $q$ in learned-qs do
    importance$[q]$ = total-impact$[q]$ / counter$[q]$
return learned-qs
```

Fig. 6. Learning of typical importances of potential questions. The procedure applies the impact-based elicitor to multiple problems, and determines the mean importance of each question.

```
Input: Question $q$ about a property of a specific room.
Output: Estimated importance of this question.

GET-IMPORTANCE($q$, room)
if $q$ has a learned importance
    then return importance$[q]$
room-type = parent[room]
while room-type is not NIL
        and the system has no learned importances of any
            similar questions for rooms of room-type do
    room-type = parent[room-type]
if room-type is NIL
    then return NIL (i.e. no importance estimate)
let similar-qs be the set of similar questions with
    learned importances for rooms of room-type
impact-sum = 0
counter-sum = 0
for each question sim-q in similar-qs do
    impact-sum = impact-sum + total-impact[sim-q]
    counter-sum = counter-sum + counter[sim-q]
return impact-sum / counter-sum
```

Fig. 7. Generalization of question importances. We give the generalization procedure for the room hierarchy, which estimates the importance of a given question by averaging the importances of other similar questions.

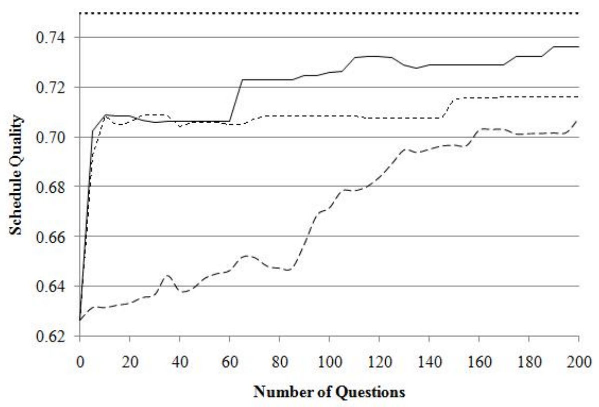

Fig. 8. Dependency of the schedule quality on the number of questions, for the elicitation based on the learned knowledge (solid line) and without learned knowledge (short dashes), as well as for the random selection of questions (long dashes). We also show the quality of the schedule constructed based on the full knowledge of all parameters (horizontal dotted line at the top).

the values of 1200 parameters are uncertain, which means that the system may potentially ask 1200 questions.

We have trained the adaptive elicitor on sixty scheduling problems, produced by the problem-generation procedure, and then used the learned knowledge in the elicitation. During the elicitation process, the system has first constructed a schedule based on the initial incomplete data; then, it has asked for additional data, five questions at a time, and modified the schedule after receiving answers to each "batch" of five questions. We have also run the analogous elicitation process without the learned knowledge, which has served as a baseline for evaluating the impact of learning.

We summarize the results in Figure 8, which shows the dependency of the schedule quality on the number of questions. We show this dependency for the elicitation based on the learned knowledge (solid line) and analogous results without learning (short dashes). We also show the results of elicitation using completely random selection of questions, without any learning or impact analysis (long dashes). The results confirm that the learning helps to improve the elicitation effectiveness, and enables the system to construct a better schedule.

VI. Concluding remarks

We have described a technique for learning the importance of potential questions about missing data, which helps to improve the elicitation of additional information in scheduling problems. Although we have applied this technique to conference scheduling, it does not rely on any specific features of this task, and it is applicable to a variety of optimization problems. We are now working on a general-purpose elicitation system, embedded into Excel, which can be integrated with a wide range of optimization architectures [12].

Acknowledgment

We are grateful to P. Matthew Jennings, Steven Gardiner, Peter Smatana, Blaze Iliev, and Ulas Bardak for their work on the scheduling and elicitation system, and for their help with the development of the described learning algorithm. We also thank Andrew Yeager for testing and evaluating this system.

References

[1] U. Bardak. "Information elicitation in scheduling problems." Ph.D. Thesis, Language Technologies Institute, Carnegie Mellon University, 2007.

[2] U. Bardak, E. Fink, and J. G. Carbonell. "Scheduling with uncertain resources: Representation and utility function." In *Proceedings of the* IEEE *International Conference on Systems, Man, and Cybernetics*, pp. 1486–1492, 2006.

[3] U. Bardak, E. Fink, C. R. Martens, and J. G. Carbonell. "Scheduling with uncertain resources: Elicitation of additional data." In *Proceedings of the* IEEE *International Conference on Systems, Man, and Cybernetics*, pp. 1493–1498, 2006.

[4] C. Boutilier, R. Das, J. O. Kephart, G. Tesauro, and W. E. Walsh. "Cooperative negotiation in autonomic systems using incremental utility elicitation." In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pp. 89–97, 2003.

[5] C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. "Constraint-based optimization with the minimax decision criterion." In *Proceedings of the Ninth International Conference on Principles and Practice of Constraint Programming*, pp. 168–182, 2003.

[6] C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. "Regret-based utility elicitation in constraint-based decision problems." In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pp. 929–934, 2005.

[7] R. D. Burke, K. J. Hammond, and B. C. Young. "Knowledge-based navigation of complex information spaces." In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 462–468, 1996.

[8] R. D. Burke, K. J. Hammond, and B. C. Young. "The FindMe approach to assisted browsing." IEEE *Expert*, 12(4), pp. 32–40, 1997.

[9] L. Chen and P. Pu. "Survey of preference elicitation methods." Swiss Federal Institute of Technology in Lausanne, Technical Report No. IC/200467, 2004.

[10] E. Fink, U. Bardak, B. Rothrock, and J. G. Carbonell. "Scheduling with uncertain resources: Collaboration with the user." In *Proceedings of the* IEEE *International Conference on Systems, Man, and Cybernetics*, pp. 11–17, 2006.

[11] E. Fink, P. M. Jennings, U. Bardak, J. Oh, S. F. Smith, and J. G. Carbonell. "Scheduling with uncertain resources: Search for a near-optimal solution." In *Proceedings of the* IEEE *International Conference on Systems, Man, and Cybernetics*, pp. 137–144, 2006.

[12] B. Fu, E. Fink, and J. G. Carbonell. "Analysis of uncertain data: Tools for representation and processing." In *Proceedings of the* IEEE *International Conference on Systems, Man, and Cybernetics*, 2008.

[13] G. Linden, S. Hanks, and N. Lesh. "Interactive assessment of user preference models: The automated travel assistant." In *Proceedings of the Sixth International Conference on User Modeling*, pp. 67–78, 1997.

[14] P. Pu and B. Faltings. "Personalized navigation of heterogeneous product spaces using SmartClient." In *Proceedings of the 2002 International Conference on Intelligent User Interfaces*, pp. 212–213, 2002.

[15] R. Patrascu, C. Boutilier, R. Das, J. O. Kephart, G. Tesauro, and W. E. Walsh. "New approaches to optimization and utility elicitation in autonomic computing." In *Proceedings of the National Conference on Artificial Intelligence*, pp. 140–145, 2005.

[16] M. Torrens, P. Herzog, L. Samson, and B. Faltings. "Reality: A scalable intelligent travel planner." In *Proceedings of the 2003* ACM *Symposium on Applied Computing*, pp. 623–630, 2003.