1-1-2006

Scheduling with Uncertain Resources Part 3: Elicitation of Additional Data

Ulas Bardak
Carnegie Mellon University, cyprus@cs.cmu.edu

Eugene Fink
Carnegie Mellon University, eugenefink@cmu.edu

Chris R. Martens
Carnegie Mellon University

Jaime G. Carbonell
Carnegie Mellon University, jgc@cs.cmu.edu

Scheduling with Uncertain Resources: Elicitation of Additional Data

Ulas Bardak, Eugene Fink, Chris R. Martens, and Jaime G. Carbonell

***Abstract*—We consider the task of scheduling a conference based on incomplete data about available resource and scheduling constraints, and describe a procedure for automated elicitation of additional data. This procedure is part of an interactive system for scheduling under uncertainty, which identifies critical missing data, generates related questions to the human administrator, and uses answers to improve the schedule.**

I. Introduction

When we work on a practical scheduling task, we usually do not have complete knowledge of the related resources and constraints. For example, when scheduling a conference, we may not know the exact sizes of available rooms or equipment needs of some speakers. The task of constructing a schedule based on incomplete data gives rise to several related problems, including the representation of uncertainty, efficient search for near-optimal schedules, and elicitation of additional data that help to reduce uncertainty.

We have explored these problems and built a scheduling system that supports the use of incomplete data. This work has been part of the RADAR project (www.radar.cmu.edu) at Carnegie Mellon University, which is aimed at building an intelligent system for assisting an office manager. We have described initial results in three earlier papers; specifically, we have explained the representation of uncertainty [Bardak *et al*., 2006a], search for a near-optimal schedule [Fink *et al*., 2006], and collaboration between the scheduling system and human administrator [Fink *et al*., 2006b].

We now present a procedure for elicitation of additional data about resources and constraints. We give a review of the related work (Section II), explain the representation of uncertainty in the developed scheduling system (Sections III and IV), describe the elicitation procedure (Sections V and VI), and give experiments on its effectiveness (Section VII).

Manuscript received on March 30, 2006. The described work has been supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010.

U. Bardak, E. Fink, C.R. Martens, and J.G. Carbonell are with Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213. The e-mail of U. Bardak is cyprus@cs.cmu.edu, the e-mail of E. Fink is e.fink@cs.cmu.edu, the e-mail of C. Martens is cmartens@andrew.cmu.edu, and the e-mail of J.G. Carbonell is jgc@cs.cmu.edu.

II. Related work

The elicitation of additional data is important in a variety of applications, and researchers investigated a number of elicitation techniques [Chen and Pu, 2004].

For example, Burke *et al*. [1996; 1997] built an assisted-browsing system that helped the user to construct her queries. Linden *et al.* [1997], Pu and Faltings [2002], and Torrens *et al*. [2003] developed systems that helped customers to find airline tickets, by eliciting information about desirable flights. The authors of these systems assumed that every query had only a few uncertain parameters, and their approach would be impractical for scheduling problems, which may have thousands of uncertain values.

Boutilier and his colleagues studied elicitation techniques for the optimization under uncertainty. They developed a procedure that generated questions about missing data, and used it in domain-independent optimization [Wang and Boutilier, 2003; Boutilier *et al*., 2003c; Boutilier *et al*., 2005], and in several domain-specific systems [Boutilier *et al*., 2003a; Patrascu *et al*., 2005]. Their procedure allowed uncertainty only in discrete parameters; thus, it would be inapplicable to scheduling with continuous values.

Researchers also developed several procedures for "collaborative filtering," which reconstructed missing preferences of a new user from the past experience with other similar users. In particular, Boutilier developed a procedure that selected questions based on the past values of the related answers [Boutilier and Zemel, 2003; Boutilier *et al*., 2003b]. Other researches applied collaborative filtering to making user-specific recommendations of news articles [Resnick *et al*., 1994], videos [Hill *et al*., 1995], music [Shardanand and Maes, 1995], and eBay products [Schafer *et al*., 2001]. The developed procedures required the users to rank relevant items; this approach would be inapplicable to large-scale scheduling problems because we cannot represent the related preferences as a ranking of standard items.

We have considered the problem of scheduling a conference based on uncertain information about resources, conference events, and scheduling constraints. It may involve uncertainty in resources, constraints, and utility functions, and its representation may contain 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.

III. SCHEDULING PROBLEM

We begin with an example of a conference scenario, and use it to illustrate the representation of resources, constraints, and preferences [Bardak *et al*., 2006]. Suppose that we need to assign rooms to events at a small one-day conference, which starts at 11:00am and ends at 4:30pm, and that we can use three rooms: auditorium, classroom, and conference room (Table 1). These rooms host other events, and they are available for the conference only at the following times:

Auditorium: 11:00am–1:30pm and 3:30pm–4:30pm.
Classroom: 11:00am–2:30pm.
Conference room: 12:00pm–4:30pm.

We describe each room by its name and a set of properties; in this example, we consider three properties:

Size: Room area in square feet.
Mikes: Number of microphones.
Stations: Maximal number of demo stations that can be set up in the room.

For every room, we define its property values (Table 1), as well as its availability, represented by a set of time intervals.

Suppose further that the conference includes five events: demonstration, discussion, tutorial, workshop, and committee meeting (Table 2). For each event, we specify its name, importance value, and related constraints and preferences. We define constraints for an event by limiting acceptable start times, durations, and room properties. For example, we may indicate that an acceptable start time for the committee meeting is 3:00pm or later, an acceptable duration is 30 minutes or more, and an acceptable room size is 400 square feet or more. We may also select preferred values for start times, durations, and room properties, which are subsets of acceptable values. For example, we may specify that the preferred start time for the committee meeting is 3:30pm, preferred duration is 60 minutes, and preferred room size is 800 square feet or more. In Table 2, we give constraints and preferences for all events.

If we do not have exact values for some room properties, importances, constraints, and preferences, then we represent every uncertain parameter as an interval, and we assume that all values in this interval are equally likely. For example, we may specify that the size of the conference room is between 500 and 750, the importance of the demo is between 4 and 6, and its minimal acceptable duration is between 60 and 90.

To build a schedule, the system assigns a specific room and time slot to every event. It represents this assignment by event name, room name, start time, and duration. Alternatively, it may decide that an event is not part of the schedule, which is also considered an assignment. In Figure 1, we show a schedule for the example problem, which satisfies all constraints and most preferences given in Table 2.

	Auditorium	Classroom	Conf. room
Size	1200	700	500
Stations	10	5	5
Mikes	5	1	2

Table 1. Available rooms and their properties.

		Demo	Discu-ssion	Tuto-rial	Com-mittee	Work-shop
Importance		5	3	8	1	5
Start time	Acceptable	Any	Any	11am	≥3pm	Any
	Preferred			11am	3:30pm	
Dura-tion	Acceptable	≥60	≥30	≥30	≥30	≥60
	Preferred	150	90	60	60	120
Room size	Acceptable	≥600	≥200	≥400	≥400	≥600
	Preferred	≥1200	≥600	≥600	≥800	≥1000
Stat-ions	Acceptable	≥5	Any	Any	Any	Any
	Preferred	≥10		≥2		
Mikes	Acceptable	Any	≥2	≥1	Any	≥1
	Preferred		≥4	≥2		≥1

Table 2. Conference events and related constraints and preferences.

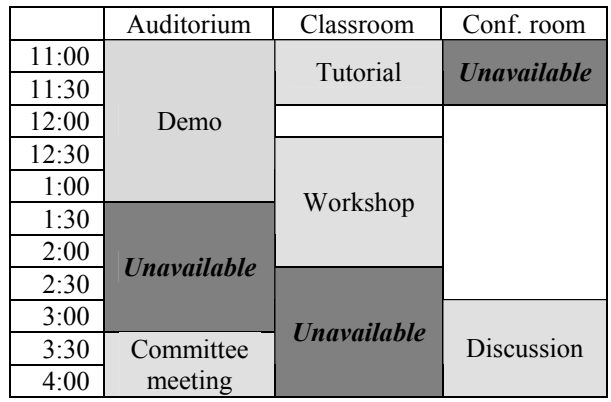

Figure 1. Schedule for the conference scenario in Tables 1 and 2.

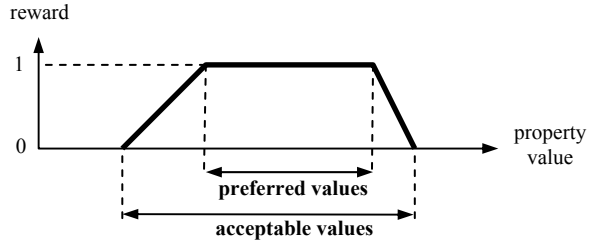

Figure 2. Reward for satisfying a preference. If the related property value is within the preferred set, the reward is 1.0; else, it linearly decreases with the distance from the set.

The procedure inputs a property value, *prop*; the minimal and maximal acceptable values for this property, *min-ac* and *max-ac*; and minimal and maximal preferred values, *min-pref* and *max-pref*.

Note that *min-ac* ≤ *min-pref* ≤ *max-pref* ≤ *max-ac*.

If the property is within the acceptable interval, the procedure returns the respective reward value; else, it returns NIL.

REWARD(*prop*, *min-ac*, *max-ac*, *min-pref*, *max-pref*)
if *min-ac* ≤ *prop* < *min-pref*
  then return (*prop* − *min-ac*) / (*min-pref* – *min-ac*)
if *min-pref* ≤ *prop* ≤ *max-pref* **then return** 1
if *max-pref* < *prop* ≤ *max-ac*
  then return (*max-ac* − *prop*) / (*max-ac* − *max-pref*)
return NIL

Figure 3. Computing the reward for satisfying a given preference.

- Provide the exact value for an uncertain room property.
  Example: Find out the size of the conference room.
- Provide the exact value for an uncertain request importance.
  Example: Find out the importance of the demo.
- Provide the exact specification for a set of acceptable values for start time, duration, or room property in an event description.
  Example: Find out the acceptable duration of the demo.
- Provide the exact specification for a set of preferred values for start time, duration, or room property in an event description.
  Example: Find out the preferred room size for the discussion.

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

IV. Schedule quality

We measure the quality of schedules on the scale from 0.0 to 1.0; higher values correspond to better schedules. The quality of a specific assignment depends on how well the selected room and time slot match the preferred values. If the start time, duration, or some room property is outside the acceptable set, then the assignment quality is zero. If an event is not part of the schedule, the assignment quality is also zero.

If an assignment satisfies all hard constraints, we determine the rewards for its preferences. If a start time, duration, or room property is within the preferred set of values, the respective reward is 1.0. If it is outside the preferred set, the reward depends on its distance from this set; specifically, the reward linearly decreases with the distance from the preferred values, as shown in Figure 2. We give pseudocode for the procedure that computes the reward value in Figure 3. If an event has k preferences, and the respective rewards are $r_1,\ldots, r_k$, then the assignment quality is $(r_1 + \ldots + r_k) / k$.

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

$$\left(imp_1 \cdot Qual_1 + \ldots + imp_n \cdot Qual_n\right) / \left(imp_1 + \ldots + imp_n\right).$$

For example, if we use the preferences in Table 2, and the schedule is as shown in Figure 1, then the assignment quality for the demo is 1.0, for the discussion is 0.75, for the tutorial is 0.8, for the committee meeting is 1.0, and for the workshop is 0.85, and the overall schedule quality is 0.86.

If the description of rooms and events includes uncertainty, we evaluate candidate schedules by the mathematical expectation of their quality. If the assignment of an event violates some hard constraint with a nonzero probability, its quality is zero. If it satisfies all hard constraints, the system computes its expected quality; specifically, it determines the expected rewards for all preferences, $E(r_1),\ldots,E(r_k)$, and uses them to compute the expected assignment quality, which is $(E(r_1) + \ldots + E(r_k)) / k$. The system uses the expected quality of assignments, along with the expected importance values, to determine the expected schedule quality, which is

$$\left(E(imp_1) \cdot E(Qual_1) + \ldots + E(imp_n) \cdot E(Qual_n)\right) / \left(E(imp_1) + \ldots + E(imp_n)\right).$$

We have given an algorithm for fast computation of this expected quality in the paper on the representation of uncertainty [Bardak *et al*., 2006].

The system searches for a schedule with a high expected quality; that is, it uses the expected quality as the utility function. The search algorithm is based on hill-climbing; it does not guarantee optimality, but is usually finds near-optimal solutions. If we apply it to construct a new schedule, it begins with the initially empty schedule and gradually improves it. If we use it to repair a schedule after changing resources or constraints, it starts with the old schedule. At each step, it either assigns a slot to some unscheduled event, or moves some scheduled event to a better slot. When the algorithm assigns a new slot, it never violates the event's constraints on the acceptable time and room properties; if it cannot find a slot that satisfies all these constraints with full certainty, it leaves the event unscheduled. The algorithm stops when it cannot find further improvements, or when reaching a time limit. We have given a more detailed description of this algorithm in the paper on the search for a near-optimal schedule [Fink *et al*., 2006b].

V. Elicitation problem

The system computes not only the expected quality of candidate schedules, but also the standard deviation of the expected quality. If this standard deviation is high, the true schedule quality may turn out much lower than the expected quality. For example, if the classroom properties are uncertain, and the schedule is as shown in Figure 1, then we risk placing the workshop and tutorial into a low-quality room.

The system may reduce uncertainty by asking the human administrator to provide more accurate data. For instance, it may ask to measure the classroom size and check the number of microphones in the classroom. We list the types of possible questions in Figure 4; every question corresponds to an uncertain value, and the number of possible questions equals the number of uncertain values.

The human effort involved in providing answers may vary from question to question. For example, checking the number of microphones is easier than measuring the room size. We represent this effort by question costs; that is, we assign different costs to different questions, and subtract the costs of all answered questions from the final schedule quality.

The purpose of the elicitation procedure is to identify a small number of critical questions, which help to improve the schedule without incurring a high elicitation cost.

VI. Estimate of question utilities

We define the utility of a question trough its potential effect on the schedule quality. Specifically, this utility is the expected increase of the schedule quality due to finding out the answer and then rescheduling.

The system estimates a question utility by the impact of the respective uncertain value on the standard deviation of the schedule quality. To determine this impact, it replaces all other uncertain parameters by their mathematical expectations, and then computes the standard deviation of the schedule quality. We show this computation for an uncertain room property in Figure 5, for an uncertain event importance in Figure 6, and for an uncertain range of acceptable values in Figure 7. We do not show the computation for an uncertain range of preferred values because it is similar to that for uncertain acceptable values in Figure 7.

For instance, consider the example in Section III, and suppose that the conference-room size is between 500 and 750, the importance of the demo is between 4 and 6, its minimal acceptable duration is between 60 and 90, and all other resources and preferences are fully certain, as shown in Tables 1 and 2. Then, the impact of the conference-room size on the standard deviation of the schedule quality is 0.00027, the impact of the demo importance is 0.026, and the impact of the minimal acceptable duration of the demo is zero.

The elicitation procedure estimates the utility of all questions and prunes the questions whose estimated utilities are no greater than their costs. Then, it sorts the remaining questions in the decreasing order of the differences between the utility and cost, and displays them to the user in this order. The system does not expect the user to provide all answers; the human administrator may answer some questions and delay or ignore the others. When she provides some answers, the system improves the schedule based on this information, re-evaluates the utility of the remaining questions, and re-orders them according to the new utility estimates.

VII. Experiments

We have applied the developed system to schedule a four-day conference, which includes eighty-four events and uses thirteen rooms; every event has seventeen preferences, and every room has fifteen properties. The representation of this problem includes 2500 parameters; the values of 700 parameters are uncertain, which means that the elicitor may potentially ask 700 questions.

First, the system has constructed a schedule based on the available incomplete knowledge; then, it has asked for additional data, one question at a time, and modified the schedule after each answer. We illustrate the results in Figure 9, which shows the dependency of the schedule quality on the number of questions. We plot the true quality, evaluated based on the complete knowledge (solid line), as well as the system's quality estimate based on its partial knowledge (dashed line).

The procedure inputs the lowest and highest possible values of an uncertain room property, *low-prop* and *high-prop*; its minimal and maximal acceptable values, *min-ac* and *max-ac*; and its minimal and maximal preferred values, *min-pref* and *max-pref*.

Note that *min-ac* ≤ *min-pref* ≤ *max-pref* ≤ *max-ac*; furthermore, for a valid schedule, *min-ac* ≤ *low-prop* < *high-prop* ≤ *max-ac*.

The procedure returns the standard deviation of the reward for satisfying the preference for the given room property.

It uses the following local variables:

low-r, *high-r*	reward values for *low-prop* and *high-prop*
prob-small	probability that the property value is below *min-pref*
prob-large	probability that the property value is above *max-pref*
exp-r	mathematical expectation of the reward value
exp-sqr-r	mathematical expectation of the squared reward value

```
LOCAL-IMPACT(low-prop, high-prop, min-ac,
             max-ac, min-pref, max-pref)
if min-pref ≤ low-prop < high-prop ≤ max-pref then return 0
low-r = REWARD(low-prop, min-ac, max-ac, min-pref, max-pref)
high-r = REWARD(high-prop, min-ac, max-ac, min-pref, max-pref)
if min-ac ≤ low-prop < high-prop < min-pref
   then return (high-r − low-r) / (2 · √3 )
if max-pref < low-prop < high-prop ≤ max-ac
   then return (low-r − high-r) / (2 · √3 )
if min-pref ≤ low-prop then prob-small = 0
else prob-small = (min-pref − low-prop) / (high-prop − low-prop)
if  high-prop ≤ max-pref then prob-large = 0
else prob-large = (high-prop − max-pref) / (high-prop − low-prop)
exp-r = small-prob · (low-r + 1) / 2 + large-prob · (high-r + 1) / 2
        + (1 − small-prob − large-prob)
exp-sqr-r = small-prob · (low-r^2 + low-r + 1) / 3
        + large-prob · (high-r^2 + high-r + 1) / 3
        + (1 − small-prob − large-prob)
return (exp-sqr-r − exp-r^2)^0.5
```

The procedure inputs the lowest and highest possible values of an uncertain room property, *low-prop* and *high-prop*; a list of the events scheduled in the room with this property, *events*; and the sum of the importances of all conference events, *sum-imps*.

It returns the impact of the uncertain room property on the standard deviation of the schedule quality.

It uses the following local variables:

std-r	standard deviation of the related reward for one event
total-sqr-std	sum of the squared standard deviations for all events, weighted by the respective importances

```
PROPERTY-IMPACT(low-prop, high-prop, events, sum-imps)
for every event in events do
   let k be the number of event's preferences,
      imp be the importance of event,
      min-ac and max-ac be the minimal and maximal
         acceptable values of the given property, and
      min-pref and max-pref be the minimal and maximal
         preferred values of this property
   std-r = LOCAL-IMPACT(low-prop, high-prop, min-ac,
                        max-ac, min-pref, max-pref)
   total-sqr-std = total-sqr-std + (imp · std-r / k)^2
return total-sqr-std^0.5 / sum-imps
```

Figure 5. Computing the impact of an uncertain room property on the standard deviation of the overall schedule quality. Note that this computation uses the REWARD procedure, given in Figure 3.

The procedure inputs an uncertain event importance, represented by its lowest and highest possible values, *low-imp* and *high-imp*; the quality of the respective event assignment, *qual*; and the sum of the importances of all conference events, *sum-imps*.

It returns the impact of the uncertain importance on the standard deviation of the schedule quality. The computation of this impact is an approximation, based on the assumption that *low-imp* and *high-imp* are significantly smaller than *sum-imps*.

IMPORTANCE-IMPACT(*low-imp*, *high-imp*, *qual*, *sum-imps*)
std-imp = (*high-imp* − *low-imp*) / (2 · $\sqrt{3}$)
return *qual* · *std-imp* / *sum-imps*

Figure 6. Impact of an uncertain importance on the schedule quality.

The procedure inputs an uncertain range of acceptable values, represented by the lowest and highest possible values of its left endpoint, *low-min-ac* and *high-min-ac*, and the lowest and highest possible values of its right endpoint, *low-max-ac* and *high-max-ac*.

It also inputs the respective range of preferred values, represented by its endpoints, *min-pref* and *max-pref*; the actual value of the respective property in the current schedule, *prop*; the number of preferences in the respective event, *k*; the importance of this event, *imp*; and the sum of the importances of all conference events, *sum-imps*.

Note that *low-max-ac* ≤ *min-pref* ≤ *max-pref* ≤ *high-max-ac*; furthermore, for a valid schedule, *high-max-ac* ≤ *prop* ≤ *low-min-ac*.

The procedure returns the impact of the acceptable-value uncertainty on the standard deviation of the schedule quality.

It uses the following local variables:
exp-r mathematical expectation of the reward value
exp-sqr-r mathematical expectation of the squared reward value

CONSTRAINT-IMPACT(*low-min-ac*, *high-min-ac*, *low-max-ac*, *high-max-ac*, *min-pref*, *max-pref*, *prop*, *k*, *imp*, *sum-imps*)
if *min-pref* ≤ *prop* ≤ *max-pref* **then return** 0
if *prop* < *min-pref*
then *exp-r* = 1 − ((*min-pref* − *prop*) / (*high-min-ac* − *low-min-ac*))
· (ln(*min-pref* − *low-min-ac*) − ln(*min-pref* − *high-min-ac*))
exp-sqr-r = 2 · *exp-r* − 1 + (*min-pref* − *prop*)^2
/ ((*min-pref* − *low-min-ac*) · (*min-pref* − *high-min-ac*))
else *exp-r* = 1 − ((*prop* − *max-pref*) / (*high-max-ac* − *low-max-ac*))
· (ln(*high-max-ac* − *max-pref*) − ln(*low-max-ac* − *max-pref*))
exp-sqr-r = 2 · *exp-r* − 1 + (*prop* − *max-pref*)^2
/ ((*high-max-ac* − *max-pref*) · (*low-max-ac* − *max-pref*))
return *imp* · ((*exp-sqr-r* − *exp-r*^{2})$^{0.5}$ / *k*) / *sum-imps*

Figure 7. Computing the impact of an uncertain range of acceptable values on the standard deviation of the schedule quality.

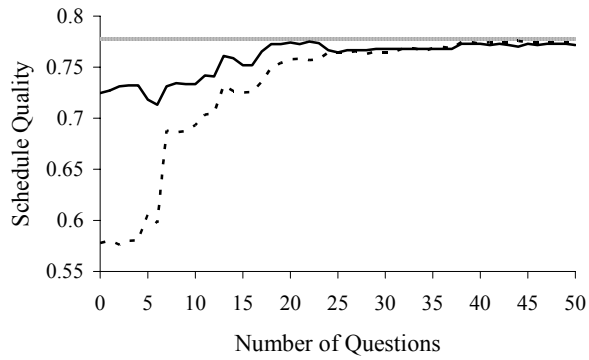

Figure 8. Dependency of the schedule quality on the number of elicitation questions. We show the true quality, computed based on the full knowledge (solid line), and the system's quality estimate based on the available incomplete data (dashed line). We also show the quality of the schedule constructed based on the full world knowledge (horizontal gray line).

The results confirm that the elicitation helps to improve the schedule; specifically, the quality has increased from 0.72 to 0.77. The system has correctly selected important questions, and asked them in the beginning of the elicitation. It has found a near-optimal schedule after asking 20 questions, which is 3% of all potential questions, whereas the remaining 680 questions have given almost no further improvement.

VIII. Concluding Remarks

We have described a procedure for identifying critical missing information about a scheduling problem; it enables the system to focus on finding out important additional facts, and disregard less important facts. We are now developing an extended system, which will optimize the use of portable equipment related to the scheduled events, and elicit related missing data. We are also working on learning techniques for automated improvement of elicitation strategies.

Acknowledgments

We are grateful to Stephen F. Smith, P. Matthew Jennings, Jean Oh, Greg Jorstad, and Daniel Cheng for their help in development of the scheduling system. We thank Jason Knichel, Konstantin Salomatin, Vijay Prakash, and Sung-joo Lim for their work on testing and evaluating this system. We also thank Aaron Steinfeld and Matt Lahut for their help in applying the system to real-world scheduling problems.

References

[Bardak *et al.*, 2006] Ulas Bardak, Eugene Fink, and Jaime G. Carbonell. Scheduling with uncertain resources: Representation and utility function. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 2006.

[Boutilier *et al.*, 2003a] Craig Boutilier, Rajarshi Das, Jeffrey O. Kephart, Gerald Tesauro, and William E. Walsh. Cooperative negotiation in autonomic systems using incremental utility elicitation. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 89–97, 2003.

[Boutilier *et al.*, 2003b] Craig Boutilier, Richard S. Zemel, and Benjamin Marlin. Active collaborative filtering. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 98–106, 2003.

[Boutilier *et al.*, 2003c] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization with the minimax decision criterion. In *Proceedings of the Ninth International Conference on Principles and Practice of Constraint Programming*, pages 168–182, 2003.

[Boutilier *et al.*, 2005] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Regret-based utility elicitation in constraint-based decision problems. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 929–934, 2005.

[Boutilier and Zemel, 2003] Craig Boutilier and Richard S. Zemel. Online queries for collaborative filtering. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

[Burke *et al.*, 1996] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. Knowledge-based navigation of complex information spaces. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, volume 1, pages 462–468, 1996.

[Burke *et al.*, 1997] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. The FindMe approach to assisted browsing. *IEEE Expert*, 12(4), pages 32–40, 1997.

[Chen and Pu, 2004] Li Chen and Pearl Pu. Survey of preference elicitation methods. In Technical Report IC/200467, pages 1–23. Swiss Federal Institute of Technology in Lausanne, 2004.

[Fink *et al.*, 2006a] Eugene Fink, Ulas Bardak, Brandon Rothrock, and Jaime G. Carbonell. Scheduling with uncertain resources: Collaboration with the user. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 2006.

[Fink *et al.*, 2006b] Eugene Fink, P. Matthew Jennings, Ulas Bardak, Jean Oh, Stephen F. Smith, and Jaime G. Carbonell. Scheduling with uncertain resources: Search for a near-optimal solution. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 2006.

[Hill *et al.*, 1995] William C. Hill, Larry Stead, Mark Rosenstein, and George W. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 194–201, 1995.

[Linden *et al.*, 1997] Greg Linden, Steve Hanks, and Neal Lesh. Interactive assessment of user preference models: The automated travel assistant. In *Proceedings of the Sixth International Conference on User Modeling*, pages 67–78, 1997.

[Patrascu *et al.*, 2005] Relu Patrascu, Craig Boutilier, Rajarshi Das, Jeffrey O. Kephart, Gerald Tesauro, and William E. Walsh. New approaches to optimization and utility elicitation in autonomic computing. In *Proceedings of the National Conference on Artificial Intelligence*, pages 140–145, 2005.

[Pu and Faltings, 2002] Pearl Pu and Boi Faltings. Personalized navigation of heterogeneous product spaces using SmartClient. In *Proceedings of the 2002 International Conference on Intelligent User Interfaces*, pages 212–213, 2002.

[Resnick *et al.*, 1994] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, pages 175–186, 1994.

[Schafer *et al.*, 2001] Ben J. Schafer, Joseph A. Konstan, and John Riedl. E-commerce recommender applications. *Data Mining and Knowledge Discovery*, 5(1/2), pages 115–153, 2001.

[Shardanand and Maes, 1995] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth." In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 210–217, 1995.

[Torrens *et al.*, 2003] Marc Torrens, Patrick Herzog, Loic Samson, and Boi Faltings. Reality: A scalable intelligent travel planner. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 623–630, 2003.

[Wang and Boutilier, 2003] Tianhan Wang and Craig Boutilier. Incremental utility elicitation with the minimax regret decision criterion. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 309–316, 2003.