Scheduling with Uncertain Resources: Representation and Utility Function

Ulas Bardak cyprus@cs.cmu.edu
Eugene Fink e.fink@cs.cmu.edu
Jaime G. Carbonell jgc@cs.cmu.edu

School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

***Abstract*—We describe the representation of uncertain knowledge in a conference-scheduling system, which may include incomplete information about available resources, conference events, and scheduling constraints. We then explain the use of this incomplete knowledge in the evaluation of schedule quality.**

I. Introduction

When we work on a practical scheduling task, we usually do not have complete knowledge of the related resources and constraints, which means that we need to account for uncertainty. For example, when scheduling conference presentations, we may not know the exact preferences of specific speakers. The need to reason under uncertainty gives rise to several related problems, including representation of uncertain knowledge, estimation of the quality of candidate solutions based on this knowledge, and efficient search for near-optimal solutions.

Although researches have long realized the importance of uncertain information in optimization problems, the related work has been limited [Chen and Pu, 2004; Sahinidis, 2004; Bidot, 2005]. For example, several researchers have built systems that ask the user to provide all missing data relevant to the task, and support only qualitative reasoning about uncertainty [Burke *et al*., 1997; Burke, 2000; Stolze and Rjaibi, 2001; Schafer *et al*., 2001; Faltings *et al*., 2004; McCarthy *et al*., 2005]. This approach is effective when a problem includes only a few uncertain facts, but it is impractical for problems with a large number of uncertain variables.

Researchers have also built systems that support reasoning with incomplete knowledge, but limit the generality of representing uncertain facts. For instance, Chajewska *et al.* [1998], Lodwick *et al*. [2001], Stolze and Ströbel [2001], and Moore [2002] have allowed uncertainty only in discrete variables. Smith *et al*. [2002], Wang and Boutilier [2003], and Boutilier *et al*. [2005] have supported uncertain constraints, but they have not allowed uncertain resources. Averbakh [2001] and Lin *et al*. [2004] have developed an uncertain representation of continuous variables that represent resources, but they have not supported uncertainty in utility functions.

We have considered the problem of scheduling a conference based on uncertain information about available resources, conference events, and scheduling constraints and preferences. It may involve uncertainty in resources, constraints, and utility functions, and it may include thousands of uncertain variables. Since the previous techniques do not provide sufficient generality for this problem, we have developed a novel mechanism for representing uncertain information, and designed a search engine that generates near-optimal schedules based on partial world knowledge.

The work on this problem is part of the RADAR project (www.radar.cs.cmu.edu) at Carnegie Mellon University, which is aimed at building an intelligent system for assisting an office manager. We have reported the results of this work in a series of four papers, including this paper. In the other three papers, we have described the developed scheduling algorithms [Fink *et al*., 2006b], automated elicitation of additional data that help to reduce uncertainty [Bardak *et al*., 2006], and collaboration between the scheduling system and human user [Fink *et al*., 2006a].

We now describe the representation of the related uncertain knowledge. First, we give an example of a scheduling scenario (Section II), and explain the encoding of available resources and scheduling constraints (Sections III and IV). Then, we describe the representation of uncertainty (Sections V and VI) and its use in evaluating schedule quality (Section VII). Finally, we compare the quality of automatically built schedules with the results of manual scheduling (Section VIII).

II. Scheduling problem

We begin with an example of a conference scenario, and use it to illustrate the representation of resources and constraints. 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

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

other events on the same day, 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 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.

The conference includes five events: demonstration, discussion, tutorial, workshop, and conference-committee meeting (Table 2). For each event, the conference committee specifies its importance, as well as constraints and preferences on its time and room properties. We construct a schedule by assigning a room and time slot to every event; we give an example schedule in Figure 1.

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

Table 1. Available rooms and their properties.

	Demo	Discussion	Tutorial	Committee	Workshop
Importance	50	30	75	10	50
Start time	Any	Any	11am	3pm–4pm	Any
Duration	≥60	≥30	≥30	≥15	≥60
Room size	≥600	≥200	≥400	≥400	≥600
Stations	≥5	Any	Any	Any	Any
Mikes	Any	≥2	≥1	Any	≥1

Table 2. Events and related constraints.

		Demo	Discussion	Tutorial	Committee	Workshop
Duration	Min	60	30	30	15	60
	Good	120	60	45	30	75
	Best	150	90	60	60	120
Room size	Min	600	200	400	400	600
	Good	1000	400	600	600	800
	Best	1200	600	800	800	1000
Stations	Min	5		0		
	Good	10	Any	1	Any	Any
	Best	15		2		
Mikes	Min		2	1		1
	Good	Any	3	1	Any	1
	Best		4	2		1

Table 3. Scheduling preferences; we specify preference functions using a three-point scheme, where the "min" value of the argument corresponds to the −5.0 quality value, "good" gives the quality of 0.0, and "best" gives the quality of 1.0.

	Auditorium	Classroom	Conf. room
11:00	Demo	Tutorial	***Unavailable***
11:30			
12:00			
12:30		Workshop	
1:00			
1:30	***Unavailable***		
2:00			
2:30		***Unavailable***	
3:00			Discussion
3:30	Committee meeting		
4:00			

Fig. 1. Example schedule.

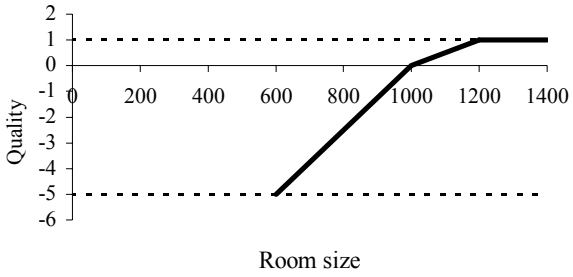


Fig. 2. Preference function, which shows the dependency of the assignment quality on the room size for the demo.

We define constraints on acceptable schedules by limiting appropriate start times, durations, and room properties for each event. For example, we may specify that an acceptable start time for the committee meeting is between 3:00pm and 4:00pm, an acceptable duration is 15 minutes or more, and a minimal acceptable room size is 400 square feet. In Table 2, we give example constraints for all five events; note that the schedule in Figure 1 satisfies these constraints.

III. Resources and Constraints

We now describe the representation of available resources, scheduling requirements, and specific schedules. We use *room* objects to represent resources; *event* objects to represent events and constraints; and *assignment* objects to represent selection of a room and time slot for each event.

Rooms: We represent a room by a name and a list of properties, such as its size, number of microphones, and the building containing it. The system allows the user to define an arbitrary list of room properties, where each property is either numeric or nominal; for example, the size of a room is a number, whereas the building that contains the room is a nominal value. For each room, we specify its availability, represented by a collection of time intervals. For instance, the auditorium in the motivating example is available for two intervals: 11am–1:30pm and 3:30pm–4:30pm.

Events: The representation of an event includes its name, importance, and constraints (see Table 2). The importance of an event is a positive integer; the higher this value, the greater the importance. The constraints are sets of acceptable values for start time, duration, and room properties.

Schedule: When the system builds a schedule, it must satisfy all hard constraints. If it violates constraints for some event, then the overall schedule is unacceptable. For example, we cannot schedule a 15-minute demo in the motivating example, even if the schedule satisfies all other constraints. To build a schedule, the system assigns a specific room and time slot to each event. It represents this assignment by four variables: event name, room name, start time, and duration. Alternatively, it can decide that an event is not part of the schedule, which is also considered an assignment. We call such an event *rejected*, and represent it internally by setting its room to NIL. Note that assignments must not overlap, that is, the system cannot assign two events to the same room at the same time. Also note that we can reject an event regardless of its constraints, since a schedule with rejected events is valid. Thus, we can always build a valid schedule by simply rejecting all events; however, its quality would be very poor.

IV. SCHEDULE QUALITY

We next define schedule quality, based on the notion of preferences, which represent soft constraints. We measure quality on the scale from −*penalty* to 1.0, where *penalty* is a nonnegative real value that represents the penalty for the worst possible schedule. Intuitively, the zero quality corresponds to satisfactory assignments, negative quality values represent poor assignments, and positive values mean unusually good assignments.

For each event, we specify preferences that represent the desirable selection of a room and time slot. We represent a specific preference by a piecewise-linear function that shows the dependency of the assignment quality on its start time, duration, or some room property. The domain of this function is the set of acceptable values for the related property, and the range is the quality values between −*penalty* and 1.0.

In Figure 2, we show an example preference, which determines the dependency of the assignment quality on the room size; in this example, the minimal acceptable room size is 600, and *penalty* is 5.0, which means that the function range is from −5.0 to 1.0. Note that this function is monotonically increasing, and we can specify it by three values of the room size, which correspond to the segment endpoints: the minimal acceptable size (600), which corresponds to the quality of −*penalty*; the satisfactory size (1000), which corresponds to the zero quality; and the minimal value of the "perfect" size (1200), which corresponds to the quality of 1.0. Although the system allows arbitrary functions, we often use this three-point scheme, which matches the human intuition. In Table 3, we give example preferences for the conference described in Section II; in this example, we use the three-point scheme to describe functions with the range from −5.0 to 1.0.

For each preference, we specify its weight, which is a positive integer that shows its relative importance compared to other preferences. In the example, we assume that all preferences have the same weight.

If we reject an event, then this assignment has the worst possible quality, which is –*penalty*. If an event has a room and time slot, we instantiate the respective start time, duration, and room properties into the event's preference functions, and take the weighted sum of their values. If an event has k preference functions, their values are $p_1,\ldots, p_k$, and their weights are $w_1,\ldots, w_k$, then the assignment quality is

$$\left(w_1 \cdot p_1 + \ldots + w_k \cdot p_k\right) / \left(w_1 + \ldots + w_k\right).$$

The overall schedule quality is the weighted sum of the quality values for individual assignments. That is, if a schedule includes n assignments, their quality values are $Qual_1,\ldots, Qual_n$, and their importances are $imp_1,\ldots, imp_n$, then the overall 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 3, and the schedule is as shown in Figure 1, then the quality of the time slot for the demo is 0.75, for the discussion is 0.75, for the tutorial is 0.62, for the committee meeting is 1.00, and for the workshop is −0.17, and the overall schedule quality is 0.50.

We use an optimization algorithm that inputs the description of rooms and events, and searches for a high-quality schedule [Fink *et al*., 2006b]. The algorithm is based on hill-climbing; it does not guarantee optimality, but it usually finds near-optimal solutions.

V. UNCERTAIN RESOUCES

When scheduling a conference, we may have incomplete data about resources, event importances, and preferences; for instance, we may not know the exact size of the conference room, or the relative importance of the demo and discussion. We represent uncertain values of room properties, event importances, and preference weights by probability density functions, approximated by collections of uniform distributions. Specifically, we encode an uncertain value by a set of disjoint intervals that may contain it, with a probability assigned to each interval; the sum of these probabilities is 1.0. In Figure 5(a), we summarize this encoding and give the related constraints on probabilities and endpoints of intervals.

For example, suppose that the exact size of the conference room is unknown. Recent measurements suggest that it is between 500 and 750, whereas old records show that it is between 1000 and 1250. If we trust the measurements more than the old records, but not completely, we may assume that the size is between 500 and 750 with 0.75 probability, and between 1000 and 1250 with 0.25 probability; in Figure 3, we show the corresponding probability density function.

VI. UNCERTAIN PREFERENCES

The representation of uncertain preferences is based on the combination of piecewise-linear functions (Section IV) with uncertain values (Section V). Specifically, we represent a preference by a piecewise-linear function that may have uncertain y-coordinates.

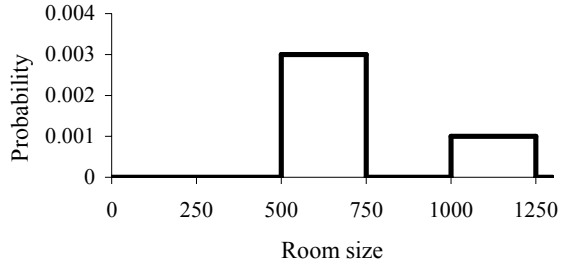
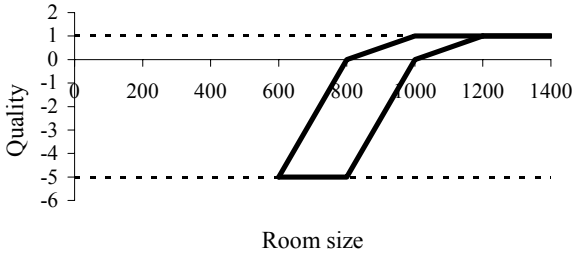
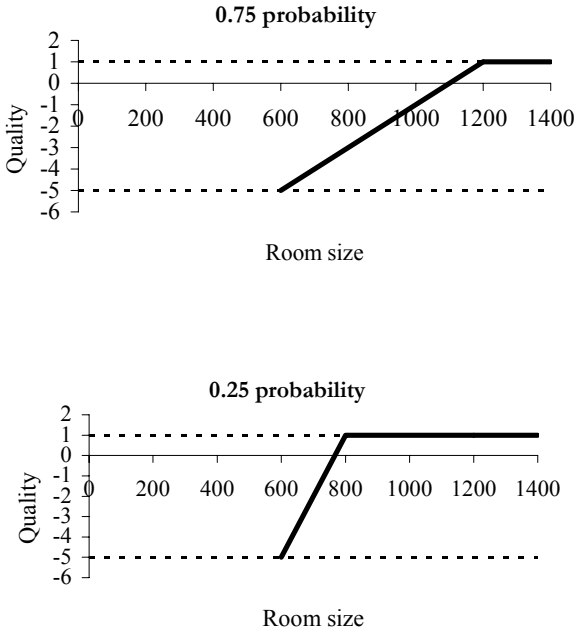


Fig. 3. Probability density function for uncertain room size, which is between 500 and 750 with 0.75 probability, and between 1000 and 1250 with 0.25 probability.



(a) Piecewise-linear function with uncertain *y*-values.



(b) Two piecewise-linear functions and their probabilities.

Fig. 4: Examples of uncertain preference functions; the first function includes uncertain quality values, whereas the second is encoded by two different functions, with probabilities of 0.75 and 0.25.

(a) Representation of an uncertain value.

$prob_1$: from min_1 to max_1
$prob_2$: from min_2 to max_2
…
$prob_m$: from min_m to max_m

We describe an uncertain value by multiple intervals and respective probabilities, and we specify each interval by its minimal and maximal value. The intervals do not overlap, and the sum of the probabilities is 1.0, which means that we impose the following constraints on the related values:

$$min_1 \le max_1 \le min_2 \le max_2 \le \ldots \le min_m \le max_m$$
$$prob_1 + prob_2 + \ldots + prob_m = 1.0$$

(b) Representation of an uncertain function.

$prob_1$: (x_{11}, y_{11}), (x_{12}, y_{12}), …
$prob_2$: (x_{21}, y_{21}), (x_{22}, y_{22}), …
…
$prob_m$: (x_{m1}, y_{m1}), (x_{m2}, y_{m2}), …

We describe an uncertain function by multiple piecewise-linear functions and respective probabilities. The description of each piecewise-linear function is a list of segment endpoints sorted by *x*-coordinate. The *x*-coordinate of a point must be a specific number, whereas its *y*-coordinate may be either a number or an uncertain value. For each piecewise-linear function, the *x*-coordinates of its endpoints are distinct:

$x_{11} < x_{12} < \ldots$
$x_{21} < x_{22} < \ldots$
…
$x_{m1} < x_{m2} < \ldots$

Furthermore, the probabilities that correspond to different piecewise-linear functions sum to 1.0:

$$prob_1 + prob_2 + \ldots + prob_m = 1.0$$

Fig. 5: Encoding of uncertain values and functions. We use uncertain values to represent room properties, event importances, and preference weights, and uncertain functions to represent preferences.

The algorithm inputs an uncertain value, represented by the vectors $prob_1 \ldots prob_m$, $min_1 \ldots min_m$, and $max_1 \ldots max_m$, as shown in Fig. 5(a).

It returns the mathematical expectation of this uncertain value.

EXPECTED-UNCERTAIN-VALUE (*prob*, *min*, *max*, *m*)
mean = 0
for *i* = 1 **to** *m* **do**
  $mean = mean + prob_i \cdot (min_i + max_i) / 2$
return *mean*

Fig. 6: Computing the mathematical expectation of an uncertain value, represented by a collection of uniform distributions and their probabilities.

For example, suppose that we need to encode a preference for a room size. Suppose further that the minimal allowed size is 600, and that 1200 is definitely enough, but we are uncertain about sizes between 600 and 1200. We believe that 800 may be an acceptable size, but there is a risk that it would be barely enough. We also believe that the size of 1000 should make the attendees perfectly happy, but there is a chance that some attendees would prefer a larger room. We may represent it by the function in Figure 4(a), where the quality for the size of 800 is an uncertain value between −5.0 and 0.0, and the quality for the size of 1000 is an uncertain value between 0.0 and 1.0.

We also allow specifying an uncertain preference by multiple functions and their probabilities. For example, suppose that some conference event requires at least 600 square feet, and the description of the event indicates that the optimal room size is 800, but a member of the conference committee has told us that the appropriate size is 1200. If we trust the committee member more than the description, but not completely, we may assume that the description is correct with probability 0.25. We then represent the size preference by two different piecewise-linear functions, with the probabilities of 0.75 and 0.25, as shown in Figure 4(b).

The developed system allows the use of uncertain quality values and multiple piecewise-linear functions at the same time; that is, we may specify several piecewise-linear functions with their probabilities, and use uncertain *y*-coordinates in each function. In Figure 5(b), we summarize the representation of uncertain preferences.

VII. Utility function

If the description of rooms and events includes uncertainty, the schedule utility depends on the mathematical expectation of the quality and on the standard deviation of the expected quality. When the system constructs a near-optimal schedule, it keeps track of the expected quality of candidate schedules. The quality computation is based on the assumption that all probability distributions are independent. If some of them are dependent, the computation does not give the exact expected quality, but it usually provides a good approximation.

If a schedule violates some hard constraint with a nonzero probability, the overall schedule quality is $-penalty$ regardless of the other constraints. If the schedule satisfies all hard constraints, the system computes the expected quality of each assignment. To estimate the assignment quality for a specific event, it determines the expected values of the related preference functions, $\mathrm{E}(p_1),\ldots, \mathrm{E}(p_k)$, as well as the expected values of their weights, $\mathrm{E}(w_1),\ldots, \mathrm{E}(w_k)$, and uses them to compute the expected quality of the assignment, which is

$$\big(\mathrm{E}(w_1)\cdot\mathrm{E}(p_1)+\ldots+\mathrm{E}(w_k)\cdot\mathrm{E}(p_k)\big)\,/\,\big((\mathrm{E}(w_1)+\ldots+\mathrm{E}(w_k)\big).$$

We give algorithms for computing the expected values of uncertain preferences and their weights in Figures 6–8. The procedure in Figure 6 finds the mean of a probability-density function represented by a collection of uniform distributions. We use it to determine the expected values of uncertain preference weights, $\mathrm{E}(w_1),\ldots, \mathrm{E}(w_k)$. The procedure in Figure 7 gives the expected value of a fully certain preference function applied to an uncertain argument, and the procedure in Figure 8 determines the expected value of an uncertain preference function. We use these procedures to compute the expected preference values, $\mathrm{E}(p_1),\ldots, \mathrm{E}(p_k)$.

The system uses the expected quality of assignments, along with the expected values of event importances, to compute the expected quality of the overall schedule, which is

$$\big(\mathrm{E}(imp_1)\cdot\mathrm{E}(Qual_1)+\ldots+\mathrm{E}(imp_n)\cdot\mathrm{E}(Qual_n)\big)\,/\,\big(\mathrm{E}(imp_1)+\ldots+\mathrm{E}(imp_n)\big).$$

It also determines the standard deviation of the schedule quality, which shows the accuracy of the quality estimate.

For instance, consider the example in Section II, and suppose that the conference-room size is represented by the uncertain value in Figure 3, the room-size preference for the demo is the uncertain function in Figure 4(a), and all other resources and preferences are fully certain, as shown in Tables 1–3. Then, the expected quality of the schedule in Figure 1 is 0.54, and its standard deviation is 0.08.

The purpose of scheduling is to increase the expected quality and reduce its standard deviation. Thus, the schedule utility monotonically increases with an increase of the expected quality, and monotonically decreases with an increase of its deviation. The developed system allows the use of an arbitrary utility function that satisfies these two conditions; it inputs a given utility function, and uses hill-climbing to construct a schedule that maximizes this function. The search algorithm begins with the empty schedule and gradually improves it; at each step, it either assigns a slot to some event that is not yet in the schedule, or moves some scheduled event to a new slot. The system continues the search until it either cannot find further improvements or reaches a time limit. We have presented more details of this algorithm in the paper on the optimization with uncertain knowledge [Fink *et al*., 2006b].

VIII. Experiments

We have applied the developed system to several scheduling problems, and compared its performance with the results of manual scheduling. Every room in these problems has fifteen properties, and every event has seventeen preference functions. We have experimented with both fully certain and uncertain world knowledge, and we have varied the number of rooms and events. We have used a 2.4-GHz Xeon computer with 400-MHz bus and 1,024-MByte memory, and set the time limit for generating schedules to 10 seconds.

In the manual-scheduling experiments, we have provided the users with a graphical interface for building a schedule, which shows room properties, event importances, constraints, and preferences [Fink *et al*., 2006a]. It allows construction of a schedule by dragging and dropping events, and it provides feedback on the schedule quality. We have not imposed any time limit on manual scheduling; most users have spent five to ten minutes on small scheduling problems, and ten to twenty minutes on large problems.

The algorithm inputs a fully certain preference function, represented by the coordinates of its endpoints, $x_1 \ldots x_l$ and $y_1 \ldots y_l$, and an uncertain value, represented by vectors $prob_1 \ldots prob_m$, $min_1 \ldots min_m$, and $max_1 \ldots max_m$, as shown in Fig. 5(a).

It returns the mathematical expectation of applying the certain preference function to the uncertain value.

We assume that all intervals of the uncertain value are in the domain of the preference function, that is, $x_1 \le min_1 \le max_m \le x_l$. If the uncertain value does not satisfy this assumption, then it violates the related hard constraint, and the system does not apply the preference function to this value.

The algorithm consists of three procedures:

- PREF-FOR-CERTAIN: Determine the value of a piecewise-linear function for a fully certain argument.
- PREF-FOR-UNIFORM: Determine the expected value of a piecewise-linear function for an uncertain argument represented by a single uniform distribution.
- EXPECTED-CERTAIN-PREF: Determine the expected value of a piecewise-linear function for an uncertain argument represented by multiple uniform distributions and their probabilities.

Input: A piecewise-linear function, represented by vectors $x_1 \ldots x_l$ and $y_1 \ldots y_l$, and a fully certain argument *arg*.

Output: The value of the function for this argument.

PREF-FOR-CERTAIN (*x*, *y*, *l*; *arg*)
Find index *a* such that either $arg = x_a$ or $x_{a-1} < arg < x_a$
if $arg = x_a$ **then return** y_a
return $(y_a \cdot arg - y_{a-1} \cdot arg + y_{a-1} \cdot x_a - y_a \cdot x_{a-1}) / (x_a - x_{a-1})$

Input: A piecewise-linear function, represented by vectors $x_1 \ldots x_l$ and $y_1 \ldots y_l$, and a uniform-distribution argument, represented by the endpoints of the distribution, *min-arg* and *max-arg*.

Output: Expected value of the function for this uncertain argument.

PREF-FOR-UNIFORM (*x*, *y*, *l*; *min-arg*, *max-arg*)
if *min-arg*=*max-arg* **then return** PREF-FOR-CERTAIN(*x*, *y*, *l*; *min-arg*)
Find index *a* such that either $min\text{-}arg = x_a$ or $x_a < min\text{-}arg < x_{a+1}$
Find index *b* such that either $max\text{-}arg = x_b$ or $x_{b-1} < max\text{-}arg < x_b$
min-y = PREF-FOR-CERTAIN (*x*, *y*, *l*; *min-arg*)
max-y = PREF-FOR-CERTAIN (*x*, *y*, *l*; *max-arg*)
if $a = b - 1$ **then return** $(min\text{-}y + max\text{-}y) / 2$
$sum = (x_{a+1} - min\text{-}arg) \cdot (min\text{-}y + y_{a+1}) / 2$
for $j = a + 1$ **to** $b - 2$ **do**
  $sum = sum + (x_{j+1} - x_j) \cdot (y_j + y_{j+1}) / 2$
$sum = sum + (max\text{-}arg - x_{b-1}) \cdot (y_{b-1} + max\text{-}y) / 2$
return $sum / (max\text{-}arg - min\text{-}arg)$

Input: A piecewise-linear function, represented by vectors $x_1 \ldots x_l$ and $y_1 \ldots y_l$, and an uncertain argument, represented by vectors $prob_1 \ldots prob_m$, $min_1 \ldots min_m$, and $max_1 \ldots max_m$, as shown in Fig. 5(a).

Output: Expected value of the function for the uncertain argument.

EXPECTED-CERTAIN-PREF (*x*, *y*, *l*; *prob*, *min*, *max*, *m*)
mean = 0
for $i = 1$ **to** *m* **do**
  $mean = mean + prob_i \cdot$ PREF-FOR-UNIFORM (*x*, *y*, *l*; min_i, max_i)
return *mean*

Fig. 7: Computing the mathematical expectation of a fully certain piecewise-linear function applied to an uncertain property value.

The algorithm inputs an uncertain preference function, represented by a collection of functions, $pref_1 \ldots pref_m$, and their probabilities, $prob_1 \ldots prob_m$. Every $pref_i$ is a piecewise-linear function, which may include uncertain *y*-coordinates, as shown in Fig. 5(b).

It also inputs an uncertain value *val*, represented by a collection of intervals and their probabilities, as shown in Fig. 5(a).

It returns the expected value of applying the uncertain preference function to the uncertain value.

EXPECTED-UNCERTAIN-PREF (*prob*, *pref*, *m*; *val*)
mean = 0
for $i = 1$ **to** *m* **do**
  for every uncertain *y*-coordinate in the representation of $pref_i$ **do**
    Call EXPECTED-UNCERTAIN-VALUE (see Fig. 6)
      to find the expected value of the uncertain *y*
    Replace the uncertain *y* in $pref_i$ with its expected value
  Call EXPECTED-CERTAIN-PREF (see Fig. 7) to find E($pref_i$(*val*)),
    which is the expected value of applying the resulting
    fully certain function $pref_i$ to the uncertain value *val*
  $mean = mean + prob_i \cdot$ E($pref_i$(*val*))
return *mean*

Fig. 8: Computing the mathematical expectation of an uncertain preference function applied to an uncertain property value.

Problem size		Schedule quality	
Number of rooms	Number of events	Manual scheduling	Automatic scheduling
Experiments with fully certain world knowledge			
5	32	0.92	0.94
9	62	0.83	0.94
13	84	0.61	0.93
Experiments with uncertain world knowledge			
5	32	0.78	0.80
9	62	0.72	0.83
13	84	0.63	0.83

Table 4: Comparison of automatic and manual scheduling.

In Table 4, we summarize the results of these experiments, which show that the system constructs better schedules than the human users, and that the difference becomes greater with an increase of the problem size.

IX. Conclusions

We have proposed a representation of incomplete knowledge in scheduling problems, which allows fast computation of expected schedule quality, and thus supports efficient search for near-optimal schedules. The proposed approach is based on representing uncertain values as probability density functions, approximated by collections of uniform distributions. We have applied it to scheduling conferences; however, it does not rely on any specific features of this task, and it is applicable to a variety of optimization problems.

The developed system has two main limitations. First, it does not support uncertainty in hard constraints; if these constraints include uncertainty, the system ensures that the probability of their violation is zero, which means that it internally replaces uncertain hard constraints with the respective worst-case constraints. Second, the system

assumes that the probability distributions of uncertain variables are independent, and it does not include a mechanism for representing dependencies among variables. We are currently working on removing these limitations.

Acknowledgments

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

References

[Averbakh, 2001] Igor C. Averbakh. On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming*, 90(2), pages 263–272, 2001.

[Bardak *et al*., 2006] Ulas Bardak, Eugene Fink, Chris R. Martens, and Jaime G. Carbonell. Scheduling with uncertain resources: Elicitation of additional data. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 2006.

[Bidot, 2005] Julien Bidot. A general framework integrating techniques for scheduling under uncertainty. PhD Thesis, Institut National Polytechnique de Toulouse, 2005.

[Boutilier *et al.*, 2004] Craig Boutilier, Tuomas Sandholm, and Rob Shields. Eliciting bid taker non-price preferences in (combinatorial) auctions. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 204–211, 2004.

[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 International Joint Conference on Artificial Intelligence*, pages 929–934, 2005.

[Burke, 2000] Robin D. Burke. Knowledge-based recommender systems. In Allen Kent, editor, *Encyclopedia of Library and Information Systems*, 69, Supplement 32. CRC Press, New York, NY, 2000.

[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.

[Chajewska *et al.*, 1998] Urszula Chajewska, Lise Getoor, Joseph Normal, and Yuval Shahar. Utility elicitation as a classification problem. In *Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 79–88, 1998.

[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.

[Faltings *et al.*, 2004] Boi Faltings, Pearl Pu, Marc Torrens, and Paolo Viappiani. Designing example-critiquing interaction. In *Proceedings of the Ninth International Conference on Intelligent Interfaces*, pages 22–29, 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.

[Lin *et al*., 2004] Xiaoxia Lin, Stacy L. Janak, and Christodoulos A. Floudas. A new robust optimization approach for scheduling under uncertainty: Bounded uncertainty. Computers and Chemical Engineering, 28(6), pages 1069–1085, 2004.

[Lodwick *et al*., 2001] Weldon A. Lodwick, Arnold Neumaier, and Francis Newman. Optimization under uncertainty: Methods and applications in radiation therapy. In *Proceedings of the Tenth IEEE International Conference on Fuzzy Systems*, pages 1219–1222, 2001.

[McCarthy *et al.*, 2005] Kevin McCarthy, James Reilly, Lorraine McGinty, and Barry Smyth. Experiments in dynamic critiquing. In *Proceedings of the Tenth International Conference on Intelligent User Interfaces*, pages 175–182, 2005.

[Moore, 2002] Frank W. Moore. A methodology for missile countermeasures optimization under uncertainty. *Evolutionary Computation*, 10(2), pages 129–149, 2002.

[Sahinidis, 2004] Nikolaos V. Sahinidis. Optimization under uncertainty: State-of-the-art and opportunities. Computers and Chemical Engineering, 28(6), pages 971–983, 2004.

[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.

[Smith *et al.*, 2002] Trey Smith, Tuomas Sandholm, and Reid Simmons. Constructing and clearing combinatorial exchanges using preference elicitation. In *Proceedings of the AAAI Workshop on Preferences in AI and CP: Symbolic Approaches*, pages 87–93, 2002.

[Stolze and Rjaibi, 2001] Markus Stolze and Walid Rjaibi. Towards scalable scoring for preference-based item recommendation. *IEEE Data Engineering Bulletin*, 24(3), pages 42–49, 2001.

[Stolze and Ströbel, 2001] Markus Stolze and Michael Ströbel. Utility-based decision tree optimization: A framework for adaptive interviewing. In *Proceedings of the Eighth International Conference on User Modeling*, pages 105–116, 2001.

[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.