Scheduling with Uncertain Resources: Collaboration with the User

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

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

***Abstract*—We describe a scheduling system that supports collaboration between the user and automated optimizer. It enables the user to monitor the optimizer decisions, make any of the decisions manually, and leave the other decisions to the system. Furthermore, it identifies the tasks that require the user's participation, and asks for assistance with these tasks.**

I. Introduction

When we work on a practical scheduling task, we often encounter unexpected changes in resources and constraints. For example, when scheduling conference presentations, we may find out that some reserved rooms are no longer available, or that some speakers have unexpected equipment needs. If these changes are significant, we may face a "crisis" situation, which requires major changes to the schedule. Furthermore, we may have to handle a continuous stream of unexpected changes, which cause multiple adjustments to the schedule before and during the conference. This need for repairing schedules in crisis situations gives rise to several related problems, including representation of uncertain knowledge, efficient automatic repairs, and support for user participation in the rescheduling process.

Although researches have long realized the importance of uncertain information in optimization problems, the related work has been limited [Chen and Pu, 2004]. 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 [Stolze and Rjaibi, 2001; Stolze and Ströbel, 2003; 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 a large number of uncertain variables.

Researchers have also studied techniques for collaboration between AI systems and users [Fleming and Cohen, 2001; Akiyoshi *et al*., 2002], including collaborative scheduling [Anderson *et al*., 2000; Ho and Lu, 2005] and planning [Allen and Ferguson, 2002; Cox and Zhang, 2005; Ai-Chang *et al*., 2004]. They have focused on domain-specific algorithms, and they have not developed techniques for general scheduling under uncertainty.

The work on graphical interfaces for collaboration between the user and automated agents has also been limited [Shneiderman and Maes, 1997]. For example, Beard *et al*. [1990], Mackinlay *et al*. [1994], and Faulring and Myers [2005] have developed systems for visualization and editing of complex schedules, which allow post-editing of automatically constructed schedules; however, they do not support interactive use of scheduling procedures.

We have explored the problem of repairing schedules in crisis situations, and built a system that helps a human manager to address sudden changes in available resources and scheduling requirements. It includes a mechanism for representing unexpected changes and related uncertainty. Furthermore, it enables the user to participate in the scheduling process, and to build a new schedule in collaboration with the system. When the user needs help, she invokes the automated scheduler; when the scheduler needs help, it asks the user for additional information or guidance. This approach allows the user to make high-level decisions and leave low-level optimization to the system.

The developed system is part of the RADAR project (www.radar.cs.cmu.edu) at Carnegie Mellon University, which is aimed at building an artificial-intelligence architecture for assisting an office manager. We have described this system in a series of four papers, including this paper. In the other three papers, we have explained the representation of uncertainty [Bardak *et al*., 2006a], search for near-optimal schedules [Fink *et al*., 2006], and automated elicitation of additional data that help to reduce uncertainty [Bardak *et al*., 2006b].

We now describe the collaboration between the system and human user. We first give an example of a scheduling scenario (Section II) and explain the encoding of available resources and scheduling constraints (Section III). We then present the architecture of the developed system (Section IV) and functionality of its main components (Sections V–VII).

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.

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 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 committee meeting (Table 2). For each event, we specify its importance, as well as related constraints and preferences. We define constraints by limiting appropriate 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.

We construct a conference schedule by assigning a room and time slot to every event. For instance, the schedule in Figure 1 satisfies all constraints and most preferences given in Table 2. The only unsatisfied preferences are the room sizes for the discussion and workshop, and the number of microphones for the discussion and tutorial.

III. Resources and constraints

We now explain the representation of available resources, scheduling requirements, and specific schedules, and then describe the utility function used in evaluation of schedules. The representation described here is a sublanguage of the full representation used in the developed system; the full language is presented in another paper [Bardak *et al*., 2006a].

Rooms: We represent resources by a set of available rooms; the description of a room includes its name and a list of numeric properties (see Table 1). The user can define an arbitrary list of properties, and then specify their values for each room. The user can also specify the availability of each room, represented by a collection of time intervals. For instance, the auditorium in the motivating example is available for 11am–1:30pm and 3:30pm–4:30pm.

Events: The description of an event includes its name, importance, and related constraints and preferences (see Table 2). The importance is a positive integer, the constraints are sets of acceptable values for start time, duration, and room properties, and the preferences are sets of preferred values.

	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		5	3	8	1	5
Start time	Acceptable	Any	Any	11am	≥3pm	Any
	Preferred			11am	3:30pm	
Duration	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
Stations	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.

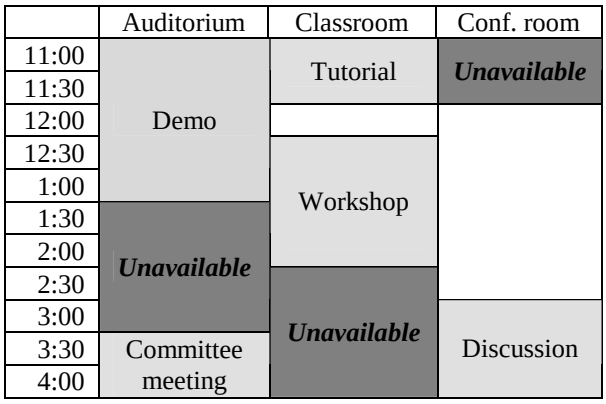


Figure 1. Schedule for the conference scenario in Tables 1 and 2.

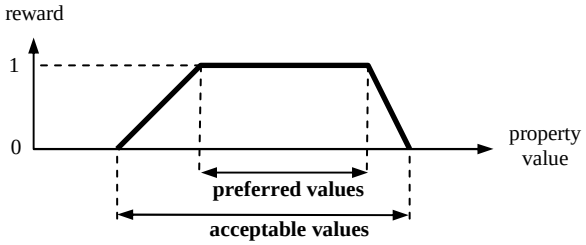


Figure 2. Reward for satisfying a preference. If the related property value is within the preferred set, the reward is 1.0; else, it linearly decreases with the distance from the set.

Uncertainty: When scheduling a conference, we may have incomplete information about resources, event importances, constraints, and preferences; for instance, we may not know the exact size of the conference room or the exact requirement for the demo duration. We represent an uncertain value as an interval, encoded by its minimal and maximal values, 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.

Schedule: 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 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.

Schedule quality: We measure quality 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 related constraints and preferences. If the start time, duration, or some room property is outside the acceptable set, then the assignment quality is zero regardless of the other constraints. For example, if we allocate a 30-minute slot for the demo in the motivating example, then the assignment quality is zero, even if it satisfies all other constraints. If we reject an event, the assignment quality is also zero.

If an assignment satisfies all hard constraints, we determine the rewards for satisfying the related 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. If the 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 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 2, and the schedule is as shown in Figure 1, then the quality of the time slot 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.

Expected quality: If the description of rooms and events includes uncertainty, we evaluate candidate schedules by the mathematical expectation of their quality [Bardak *et al*., 2006a]. The system determines the expected quality of individual assignments, $E(Qual_1),\ldots, E(Qual_n)$, as well as the expected values of their importances, $E(imp_1),\ldots, E(imp_n)$, and uses them to compute the expected quality of the overall schedule, which is

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

The quality computation is based on the assumption that the uncertain values are uniformly distributed in their respective intervals, and that these distributions are independent. If the uncertain values do not satisfy these two assumptions, the computation does not give the exact expected quality, but it usually provides a good approximation.

IV. Architecture

The scheduling architecture consists of the components shown in Figure 3. We briefly describe the role of these components; then, in Sections V–VII, we explain how they support the collaboration with the user. We have given a more detailed description of these components in the papers on the representation of uncertainty [Bardak *et al*., 2006a], scheduling based on uncertain knowledge [Fink *et al*., 2006], and elicitation of additional data [Bardak *et al*., 2006b].

World model [Bardak *et al*., 2006a]: This component maintains the description of a scheduling scenario, which includes the information about resources, constraints, and current schedule. It keeps a persistent copy of the world model on disk, and a fast-access copy in memory.

Scorer [Bardak *et al*., 2006a]: The scoring module is a fast procedure for evaluating schedule quality, which computes the expected quality of each assignment. The system uses it for automatic scheduling and for feedback during manual scheduling.

Scheduler [Fink *et al*., 2006]: The scheduling module inputs the description of rooms and events, and searches for a schedule with a high expected quality. The search algorithm is based on hill-climbing; it does not guarantee optimality, but is usually finds near-optimal solutions. If we apply this algorithm 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. It continues the search until it cannot find further improvements, or until reaching a time limit.

Elicitor [Bardak *et al*., 2006b]: The elicitation module determines whether the scheduler needs manual help, and generates respective requests to the user. We list the request types and give example requests in Figure 4.

If more accurate information about some uncertain value may help to improve the schedule, the system asks the user to find out this information. If the constraints for some event include a lot of uncertainty, the system asks the user to schedule this event manually, which is usually easier than providing all related constraints.

For each potential request, the elicitor analyzes the expected schedule improvement due to the user's help, and the expected human effort of addressing this request; it evaluates

the utility of a request by the difference between the expected improvement and the required effort. The system selects the requests with positive utility, ranks them from the highest to the lowest utility, and displays them in this order.

Top-level control: This module coordinates the invocation of the other modules, and it also routes data among them. Currently, it uses simple control procedures; we are now working on a more intelligent version, which will include heuristics and learning mechanisms for making the best use of the search algorithm, and for improving its co-ordination with the manual scheduling.

Interface: The graphical user interface consists of three main screens, as shown in Figure 3. The first screen is for editing the description of rooms and events (Section V), the second is for constructing and repairing conference schedules (Section VI), and the third is for keeping track of the requests generated by the elicitor (Section VII).

V. Editing resources and constraints

The first screen is for modifying the description of available resources and scheduling constraints; it supports the eight types of modifications summarized in Figure 5. We show the sub-screen for editing resources in Figure 7(a), and the sub-screen for constraints and preferences in Figure 7(b).

The user needs to update the world model when she learns about unexpected changes. For instance, if the conference room in the motivating example suddenly becomes unavailable because of unexpected repairs, the user must either completely delete it or mark the repair period as unavailable. As another example, if the conference committee decides to move the demo to the afternoon, the user has to adjust its set of acceptable times.

The user may also update the world model when she finds out more details about rooms or constraints. For instance, suppose that the conference-room size is uncertain, and the user needs its exact value for constructing a schedule. Then, she may measure the size of this room and input its value.

When the user updates the world model, she needs to make related modifications to the schedule. She may use the system to repair the schedule automatically; alternatively, she can build a new schedule herself in collaboration with the system.

VI. Collaborative scheduling

The scheduling screen is the central part of the interface (see Figure 8); it allows the user to invoke the automated scheduler, guide its search, and make manual modifications.

Available information: The scheduling screen provides all data about rooms and events; it allows the user to view the properties and availability of every room, importances of events, and constraints and preferences for each event. It also provides a graphical view of the schedule, and shows the reward for each preference, the quality of each assignment, and the overall schedule quality.

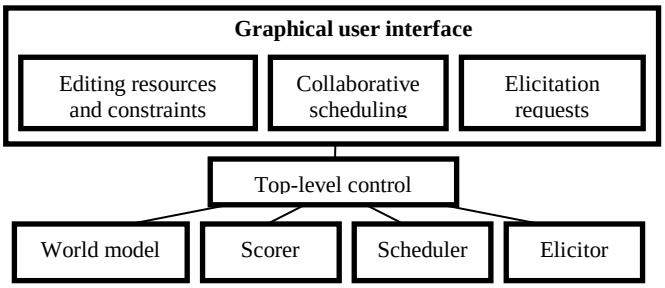


Figure 3. Architecture of the scheduling system.

- 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 event 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.
- Select a room and time for an event.
  Example: Select a time slot for the workshop.

Figure 4. Types of requests to the user. The system may ask the user to find out more information about available resources and scheduling constraints, and to schedule some events manually.

- Add a new room and specify its properties and availability.
- Delete an old room.
- Change properties of a room.
- Change the availability of a room.
- Add a new conference event and specify acceptable and preferred values for its start time, duration, and room properties.
- Delete an old event.
- Modify sets of acceptable values for an event.
- Modify sets of preferred values for an event.

Figure 5. Interface operations for editing resources and constrains.

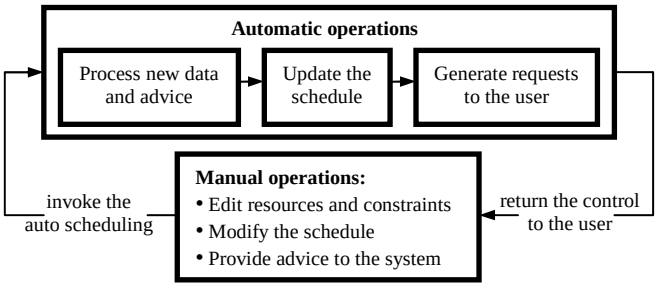
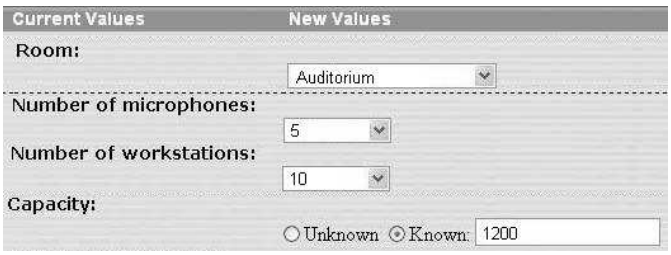
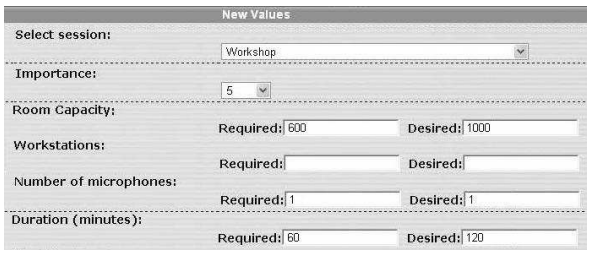


Figure 6. Main steps of the collaborative scheduling.



(a) Input of room properties.



(b) Input of event constraints and preferences.

Figure 7. Screen for editing resources and constraints.

Figure 8. Screen for collaborative scheduling.

Manual scheduling: The user can construct or modify a schedule by dragging conference events to appropriate time slots in the graphical view. She can also remove some events from the schedule by dragging them into the bin of rejected events. The system continuously recomputes the expected quality of the schedule, and shows the impact of each manual change on the quality of each assignment.

Automated scheduling: The user may invoke the search procedure that automatically improves the schedule, and she may interleave manual and automated scheduling. For instance, she can call the procedure to build an initial schedule, then make manual modifications, and then call it again to improve the schedule.

When the user invokes the scheduler, she can provide several types of restrictions on the search process. First, she can mark the rooms that should be used for the conference, and then the system will place events only into these rooms. For example, she may specify that the system should use only the auditorium and classroom.

Second, she can "lock" some events in their current locations, and then the system will not move these events. For instance, she may specify that the demo and tutorial should be scheduled as shown in Figure 1, but the system can move the other three events.

Third, the user can lock a room selected for an event without locking time; for example, she may specify that the demo must remain in the auditorium, but the system can change its start time and duration. Alternatively, the user can lock the start time of an event without restricting its location; for instance, she may indicate that the tutorial has to start at 11am, but the system can move it to another room.

Auxiliary operations: The interface also provides several auxiliary operations, which include saving the current schedule in a file, reading a previously saved schedule, undoing and redoing recent changes, and customizing the view of the schedule.

VII. Elicitation requests

The elicitation screen shows the requests for manual help, sorted in the decreasing order of their utility. It allows the user to change request priorities, mark the completed requests, and delete the requests that she does not plan to address.

The display of requests includes links to the related parts of the other screens; for example, a request to find out the exact value of a room property has a link to this property on the room-editing screen, and a request to select a time slot for an event has a link to this event on the scheduling screen.

Note that the system does not expect the user to provide all requested help. The user may address some requests and delay or ignore the others, and the system constructs the schedule based on the resulting information. If the user later addresses other requests, the system updates the schedule to account for the additional information.

VIII. Conclusions

We have described a system that helps a human manager to construct a conference schedule in a crisis situation, which may involve major unexpected changes, as well as incomplete information about resources and scheduling requirements. It allows the user to participate in the scheduling process; that is, the user can monitor the system's decisions, make any of the decisions manually, and leave the other decisions to the system. Furthermore, the system identifies the tasks that require manual help, and asks for assistance with these tasks.

In Figure 6, we summarize the steps of the collaborative scheduling process. When the user invokes the scheduling procedure, it processes the new data and advice, modifies the schedule based on these data, and generates requests to the user. It then returns the control to the user, who can update resources and constraints, modify the schedule, address the system's requests, and provide additional advice.

Although we have considered a scheduling problem, the underlying framework does not rely on specific features of this problem, and it is applicable to a variety of optimization tasks.

The developed system has two main limitations. First, the system does not provide explanation of its scheduling decisions, which reduces the effectiveness of its collaboration with the user. Second, the elicitation is limited to the five types of requests in Figure 4; the system does not consider other types of user help, such as obtaining additional resources, requesting the conference organizers to relax constraints, or anticipating possible future changes. We plan to develop an extended version of the system, which will include an explanation mechanism and a new module for negotiating additional resources and constraint changes.

Acknowledgments

We are grateful to Brad A. Myers and Andrew Faulring for their help in developing the graphical user interface. We thank Konstantin Salomatin, P. Matthew Jennings, Chris P. Martens, Jason Knichel, and Vijay Prakash 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

[Ai-Chang *et al*., 2004] Mitchell Ai-Chang, John L. Bresina, Leonard Charest, Adam Chase, Jennifer Cheng-jung Hsu, Ari K. Jonsson, Bob Kanefsky, Paul H. Morris, Kanna Rajan, Jeffrey Yglesias, Brian G. Chafin, William C. Dias, and Pierre F. Maldague. MAPGEN: Mixed-initiative planning and scheduling for the Mars Exploration Rover mission. *IEEE Intelligent Systems*, 19(1), pages 8–12, 2004.

[Akiyoshi *et al*., 2002] Masanori Akiyoshi, Teruhiko Teraoka, Yoshio Ichida, and Takayuki Yamaoka. Mixed-initiative on human-agent interaction. In *Proceedings of the Forty-First SICE Annual Conference*, volume 3, pages 1657–1661, 2002.

[Allen and Ferguson, 2002] James Allen and George Ferguson. Human-machine collaborative planning. In *Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space*, 2002.

[Anderson *et al*., 2000] David Anderson, Emily Anderson, Neal Lesh, Joe Marks, Brian Mirtich, David Ratajczak, and Kathy Ryall. Human-guided simple search. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 209–216, 2000.

[Bardak *et al*., 2006a] 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.

[Bardak *et al*., 2006b] 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.

[Beard *et al*., 1990] David Beard, Murugappan Palaniappan, Alan Humm, David Banks, Anil Nair, and Yen-Ping Shan. A visual calendar for scheduling group meetings. In *Proceedings of the Third Conference on Computer Supported Cooperative Work*, pages 279–290, 1990.

[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.

[Cox and Zhang, 2005] Michael T. Cox and Chen Zhang. Planning as mixed-initiative goal manipulation. In *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling*, pages 282–291, 2005.

[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 User Interfaces*, pages 22–29, 2004.

[Faulring and Myers, 2005] Andrew Faulring and Brad A. Myers. Enabling rich human-agent interaction for a calendar scheduling agents. In *Proceedings of the 2005 Conference on Human Factors in Computing Systems*, pages 1367–1370, 2005.

[Fleming and Cohen, 2001] Michael Fleming and Robin Cohen. A user modeling approach to determining system initiative in mixed-initiative AI systems. In *Proceedings of the Eighth International Conference on User Modeling*, pages 54–63, 2001.

[Fink *et al*., 2006] 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.

[Ho and Lu, 2005] Ken Ka Lun Ho and Meiliu Lu. Web-based expert system for class schedule planning using JESS. In *Proceedings of the 2005 IEEE International Conference on Information Reuse and Integration*, pages 166–171, 2005.

[Mackinlay *et al*., 1994] Jock D. Mackinlay, George G. Robertson, and Robert DeLine. Developing calendar visualizers for the information visualizer. In *Proceedings of the Seventh ACM Symposium on User Interface Software and Technology*, pages 109–118, 1994.

[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.

[Shneiderman and Maes, 1997] Ben Shneiderman and Pattie Maes. Direct manipulation versus interface agents. *Interactions*, 4(6), pages 42–61, 1997.

[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, 2003] Markus Stolze and Michael Ströbel. Dealing with learning in eCommerce product navigation and decision support: The Teaching Salesman Problem. In *Proceedings of the Second Interdisciplinary World Congress on Mass Customization and Personalization*, 2003.