Availability Bars for Calendar Scheduling

Andrew Faulring
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15217 USA
faulring@cs.cmu.edu

Brad A. Myers
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15217 USA
bam@cs.cmu.edu

Abstract
Calendar scheduling is a difficult task for people who have overbooked calendars with many constraints. Currently, calendar applications do not allow users to specify scheduling constraints such as how preferable a free time is for scheduling a new meeting or to what extent an existing meeting can be rescheduled. This paper introduces the "availability bar," an interaction and visualization technique for complex calendar scheduling constraints. Availability bars, embedded in calendar applications, can help users who manually schedule meetings. Availability bars can also mediate communication with calendar scheduling agents that gather availability constraints, search for times that satisfy the constraints, and negotiate with invitees when no satisfactory time is found for the constraints.

Keywords
Constraints, calendar scheduling, visualization techniques, intelligent user interfaces, agents.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces; H.4.1 [Information Systems Applications]: Office Automation---Time management.

Introduction
People perform many problem-solving tasks, such as scheduling meetings, arranging travel, and making

purchasing decisions, that require searching for solutions that satisfy complex constraints. Constraints define preferences on the solution's properties and the relative importance among all of the constraints. We are investigating constraint specification and visualization in the context of calendar scheduling.

Existing shared and non-shared calendar applications do not represent complex availability preferences for busy professionals such as managers and university faculty, making it difficult for meeting organizers to solve calendar scheduling tasks. While shared tools support collecting each invitee's free and busy times, they do not allow invitees to specify implicit, individual preferences such as how early in the day they are willing to meet, when they prefer to eat lunch, and so forth [7]. Despite their benefits, shared tools are not always available since they usually require a centrally managed server, for example a Microsoft Exchange Server. Some organizations choose not to operate such servers, and those that do generally limit access to their members, rendering the tools useless when scheduling meeting with people outside of the organization. Falling back to non-shared calendar tools forces the meeting organizer to collect availability constraints, generally via email, from each invitee, who upon receiving such a request must find times in their calendar when they are available. Email forces both people to convert availability constraints between natural language and graphical calendar representations, a tedious and error prone process.

Neither type of calendar system supports other important subtasks. First, when no common free time exists, the meeting organizer might try to negotiate with the invitees to determine if any of them can adjust their availability, perhaps by rescheduling a meeting. Second, given the choice among multiple possible meeting times, the meeting organizer has no simple way to determine how well each option satisfies the complex constraints. Third, the meeting organizer may need to modify the relative importance among the constraints. For example, some invitees' availability constraints may carry a stronger weight because their attendance is more important or they are less likely to rearrange their calendar to accommodate a new meeting. Finally, just keeping track of all the constraints and the status of different negotiations can be taxing for the meeting organizer, particularly when numerous meetings are being scheduled concurrently. Intelligent agent research is trying to solve some of these problems [5], but people will still need to specify and understand all the constraints.

The Availability Bar

We developed the "availability bar" visualization to address many of the problems with calendar applications described above. An availability bar shows how a user's preference for new meetings varies over the course of a day. Each region of time in the availability bar is assigned a preference level drawn from a continuum of values ranging between "required" and "unacceptable." The "required" level indicates that only the specified time is allowed, which is only meaningful in response to a specific request for a meeting. The "unacceptable" level indicates that the specified time is not allowed.

Graphically, the continuum is represented by a gradient of a single hue as seen in Figure 1.1. The "required" extreme is assigned a very saturated value of the hue, which should draw attention to it. The "unacceptable"

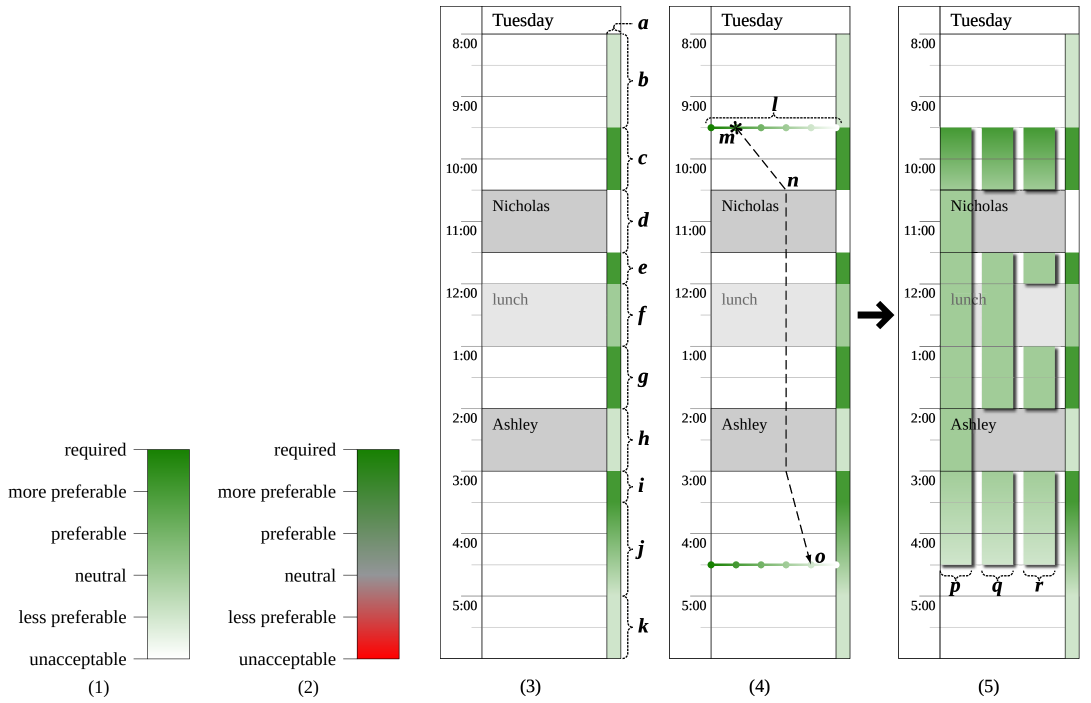

Figure 1. (1): A gradient of a single hue for encoding the preference level continuum. (2): An earlier design with a multi-hue gradient. (3): The availability bar (*a*) shows a user's availability over the course of a day (*b*–*k*). (4): In a single gesture (*m*→*n*→*o*) the user can designate a region and apply a possibly time-varying availability preference level to it. (5): The original availability preference region (*p*) and two system-generated alternatives (*q* and *r*).

extreme is assigned white (completely unsaturated). For the intermediate values, the saturation decreases proportionally with the decreasing preferability level. Figure 1.2 shows an earlier design for the preference level visualization, which used a multi-hue gradient: green ("required") to gray ("neutral") to red ("unacceptable"). When a user is looking for good times to schedule a meeting, red draws attention to the unacceptable times. Instead red is used to indicate when either a required or unacceptable constraint is violated. The single hue gradient also works better in black-and-white.

Figure 1.3 shows a user's calendar with an availability bar on the right (*a*). The less saturated ("less preferable") regions before 9:30 (*b*) and after 5:00 (*k*) represent the user's preference to not have meetings early in the morning or late in the afternoon. The unscheduled regions between 9:30 and 3:30 (*c*, *e*, *g*, and *i*), represent "more preferable" times for scheduling meetings. The meeting with Nicholas is important and cannot be easily rescheduled so the availability bar has the "unacceptable" preference level (*d*). The meeting with Ashley can be rescheduled if necessary, so the availability bar has the "less preferable" preference level (*h*). Lunch time is assigned the "neutral" preference level indicating that the user could meet during that time if necessary (*f*). Regions may also have a non-constant preference level. The preference level for the region from 3:30 to 5:00 decreases continuously from "more preferable" to "less preferable," indicating a decreasing preference for meetings during that range of time (*j*).

When necessary, the user can manually create, resize, and delete regions of the availability bar, and change the preference level for each region. We envision several mechanisms for setting the availability bar that would relieve the user of the need to do so for each day. The user may specify default availability preferences on a daily, weekly, or monthly basis that are automatically applied to each day. An intelligent agent could also learn the user's general time of day preference, which it would set as the daily default [6].

The calendaring system could use rules to update the availability bar as meetings are added, moved or deleted. For example, a rule might set the availability during meetings with one's boss to "unavailable." An intelligent agent might create rules based upon learned knowledge of the extent to which an existing meeting can be rescheduled [5].

In some situations a user needs to specify availability preferences without modifying the availability bar: for example when responding to an email containing a meeting request or when answering an agent's question. Accordingly, we developed the "painting availability" technique (see Figure 1.4 and 1.5). A user enters the "painting availability" mode by selecting a paint availability tool or by issuing a command. A horizontal availability preference level bar appears at the current mouse position with dots corresponding to each preference level, decreasing from left to right (*l*). The user initiates painting a region by clicking on the availability preference level bar, which selects the preference level at the start of the region (*m*). Dragging the mouse allows the user to specify their preference level over time; the dashed line shows the path of the mouse. While dragging the mouse, the user can change the preference level by moving horizontally and clicking the mouse (*n*). The preference level is linearly interpolated between mouse clicks. To stop painting the region, the user double clicks the mouse (*o*). The mouse stays in "availability paint" mode allowing the user to immediately paint another region. The user can edit these regions: changing start and end times, editing the availability preference levels, deleting them, and copying them across days. Once the user exits the "painting availability" mode, the display in Figure 1.5 appears with three availability options each drawn as a set of translucent bars with drop shadows (*p*, *q*, and *r*). The left option (*p*) is exactly what the user dragged out, ignoring the availability bar for that day. The calendar application may also suggest alterative options (*q* and *r*). The middle option (*q*) is created by removing any scheduled meetings from (*p*). By offering this option, the user can quickly drag out a region, knowing that scheduled meetings can be easily excluded. The rightmost option (*r*) further excludes times that are unscheduled, but not available, such as lunch. The user can edit the options and if necessary choose one to complete their response. If the user is responding to an email, the response would be copied to the clipboard for pasting into a reply email.

Making Scheduling Decisions

We extended the availability bar technique to help a meeting organizer find a time and a place for a group meeting. An availability bar for each invitee is displayed side-by-side allowing a quick visual inspection to find promising times for the meeting to be scheduled.

In the following scenario, Owen Harris, the meeting organizer, wants to schedule a lab meeting sometime during the next three days with nine other people. Figure 2 shows Owen's display after all of the invitee's availability constraints have been collected. To the right of each day appear two sets of availability bars: the availability preference of each invitee (*a* and *b*: green hue) and the availability of conference rooms capable of holding ten people (*c* and *d*: blue hue). Note that people can have different availability preference levels, hence the different shades of green, whereas rooms are either available or unavailable: blue or white, respectively.

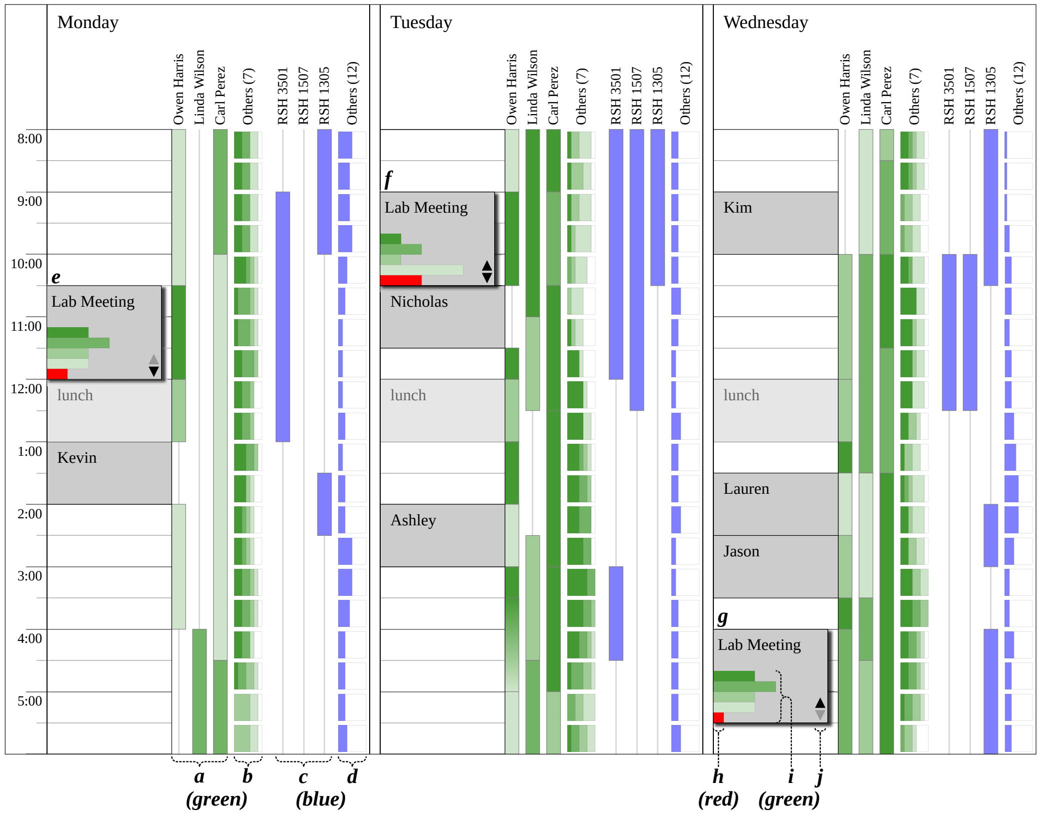

Figure 2. The display shows availability constraints of all invitees and the availability of conference rooms for a lab meeting. Three possible meeting times are shown (*e*, *f*, and *g*). At the bottom of each meeting, a histogram (*i*) shows the number of invitees per availability preference level. The red bar (*h*) draws attention to meeting options for which at least one invitee is unavailable.

Displaying the availability bars for all ten people might require too much space, so the design displays availability bars for the three most important people (*a*) and a histogram summarizing the availability of the other seven people (*b*). The histogram shows the distribution of availability levels and offers more information than simply showing an availability bar with an average or maximum value. The user can directly manipulate how many availability bars are visible. The user can change the order of availability bars, where the ordering corresponds to each bar's priority. The user can group availability bars into priority classes. For example, it might be important that all invited faculty attend the meeting, but less important that their students attend. In such a case, the faculty would be in a group with a higher priority than that of the student group. The priorities might be reversed when dealing with students who cannot reschedule their classes.

Figure 2 shows three possible times for a "Lab Meeting" (*e*, *f*, and *g*). These times were chosen for illustrative purposes; an optimizing algorithm might propose better times. Each meeting appears in the foreground layer with a drop shadow to designate that it is proposed or tentative. The histogram at the bottom of each meeting (*i*) shows the number of people in each preference level. For the "unacceptable" level, the histogram uses a red bar (*h*) instead of a white bar to draw attention to the fact that some invitees cannot attend the meeting. Highlighting one of the meetings also highlights the other options for that meeting. The arrows at the bottom of each meeting (*j*) allow the user to move the focus to the chronologically previous or next option of the same meeting and also show the user if there are options beyond the range of the current display.

Related Work

Related work falls into two categories: collaborative human-agent optimization and calendar visualizations. The human-guided simple search (HuGSS) framework explores how a human can guide an optimization algorithm by suggesting paths that look the most promising [1]. Our system focuses on collecting, visualizing and prioritizing complex constraints, whereas with HuGSS the user is not involved in defining the constraints. Another system, VEIL, asks the user to choose among alternative outcomes rather than explicitly describe their constraints [3]. The system's incremental utility elicitation (IUE) algorithm selects alternatives that should offer the most information.

The Visual Scheduler [2] and Time Lattice [4] systems visualize common free and busy times across multiple people's calendars. Both systems only support binary free-busy availability constraints, unlike our design's continuum of availability preference levels. Similar shortcomings occur in commercial shared calendar systems such as the Microsoft Exchange Server.

Conclusions and Future Work

We showed how to embed constraint specification and visualization within direct-manipulation style calendar applications. This novel approach contrasts with traditional AI techniques that use natural language dialogue interfaces for eliciting constraints. Additionally, our visualization techniques should also be useful even without an optimization algorithm.

We are presently working on designs to allow the user to specify the relative priority among constraints and to support scheduling recurring meetings. We will test our designs in lab studies to evaluate how well people can use them and to quantify performance improvements for common tasks. Then, we plan to build a calendar system incorporating our designs and deploy it within our organization to learn how the designs work in the real world, gathering information not available in lab usability studies. Finally, we also plan to apply our techniques to other domains such as arranging travel and making purchasing decisions.

Acknowledgements

The authors thank Agata Adamowicz, Justin Cinicolo, Andrew Ko, Adam Lovrovich, Jean Oh, Yeming Shi, Stephen Smith, and John Zimmerman. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010.

References

[1] Anderson, D., Anderson, E., Lesh, N., Marks, J., Mirtich, B., Ratajczak, D. and Ryall, K. Human-Guided Simple Search. *Proc. AAAI 2000*, AAAI Press (2000), 209–216.

[2] Beard, D., Palaniappan, M., Humm, A., Banks, D., Nair, A. and Shan, Y.-P. A Visual Calendar for Scheduling Group Meetings. *Proc. CSCW 1990*, ACM Press (1990), 279–290.

[3] Blythe, J. Visual Exploration and Incremental Utility Elicitation. *Proc. AAAI 2002*, AAAI Press (2002), 526–532.

[4] Mackinlay, J.D., Robertson, G.G. and DeLine, R. Developing Calendar Visualizers for the Information Visualizer. *Proc. UIST 1994*, ACM Press (1994), 109–118.

[5] Modi, P.J., Veloso, M., Smith, S. and Oh, J. CMRadar: A Personal Assistant Agent for Calendar Management. In *Agent-Oriented Information Systems II*, Springer-Verlag, 2005. 169–181.

[6] Oh, J. and Smith, S.F. Learning User Preferences in Distributed Calendar Scheduling. *Proc. PATAT 2004*, 35–50.

[7] Palen, L. Social, Individual and Technological Issues for Groupware Calendar Systems. *Proc. CHI 1999*, ACM Press (1999), 17–24.