groupTime: Preference-Based Group Scheduling

Mike Brzozowski[1], Kendra Carattini[2], Scott R. Klemmer[1], Patrick Mihelich[2], Jiang Hu[3], and Andrew Y. Ng[2]
[1] Stanford University HCI Group [2] Stanford University AI Lab [3] Stanford Dept. of Communication
353 Serra Mall, Stanford, CA 94305, USA 450 Serra Mall, Stanford, CA 94305
{zozo, kendrajc, srk, mihelich, ang}@cs.stanford.edu huj@stanford.edu

ABSTRACT

As our business, academic, and personal lives continue to move at an ever-faster pace, finding times for busy people to meet has become an art. One of the most perplexing challenges facing groupware is effective asynchronous group scheduling (GS). This paper presents a lightweight interaction model for GS that can extend its reach beyond users of current group calendaring solutions. By expressing availability in terms of *preferences*, we create a flexible framework for GS that preserves *plausible deniability* while exerting social pressure to encourage honesty among users. We also propose an ontology that enables us to model user preferences with machine learning, predicting user responses to further lower cognitive load. The combination of visualization/direct manipulation with machine learning allows users to easily and efficiently optimize meeting times. We also suggest resulting design implications for this class of intelligent user interfaces.

Author Keywords

Machine learning, supervised learning, intelligent user interfaces, group scheduling, group calendaring

ACM Classification Keywords

H5.3. Information interfaces and presentation (*e.g.*, HCI): Group and Organization Interfaces. K.4.3. Organizational Impacts: Computer-supported collaborative work.

INTRODUCTION

Ever since the advent of passenger rail spurred the adoption of Greenwich Mean Time and established a coordinated regular schedule [35], modern society has become obsessed with allocating the precious resource of time. Schedules today act as mediators between people [26], allowing them to manage their time and barter it in transactions. People ask if they could "have" each other's time, and think of how they "spend" or "waste" their time. A busy (or ostensibly busy) schedule also acts as a scapegoat, allowing its owner to blame it rather than declining a meeting directly [26].

CHI 2006, April 22-28, 2006, Montréal, Québec, Canada.
Copyright 2006 ACM 1-59593-178-3/06/0004...$5.00.

People use calendar artifacts as *memory prostheses* for events and tasks [23, 26]. A calendar serves as a "world-word" [30] mapping, by *describing* a fixed schedule (*e.g.*, "September 5 is Labor Day"), and as a "word-world" mapping, by *prescribing* things that should occur (*e.g.*, "Pay bills"). However, items on a calendar do not always directly translate to actual activity [36].

In the context of group scheduling (GS), calendars serve as communication tools; a form of "distributed cognition" [20]. Finding a time that a group of people can meet together is often aided by some expression of each participant's calendar, whether in spoken dialogue, email or instant messaging text, or in some visual representation.

Current Group Calendaring Systems

Traditional group calendaring systems (GCS) such as Microsoft Outlook and Lotus Notes present an explicit representation of users' schedules (typically whether they are free or busy) [3, 5]. For a group of users, finding a time to meet is simply a matter of choosing a time that all users appear to be free.

Yet, this binary view of availability is often inadequate to describe users' actual *preferences*. Palen's research found that scheduling has come to be viewed as "less an 'optimizing' task and more often a 'satisficing' task" [27]. As a result, suboptimal meeting times are selected. Worse, people feel compelled to pollute their calendars with misinformation to avoid appearing "free" at times they'd really rather not meet, employing "defensive scheduling" [26].

While these systems are prevalent, at least in workspaces around the world, GCS is considered "the least useful groupware application" [16]. Top among users' explicit concerns with such systems are privacy and the "prisoner's dilemma" that since such systems rely on complete knowledge of a user's schedule, they are only useful if everyone's schedule is accurate [16].

Another system that supports limited group scheduling is Evite [11], which allows a meeting's invitees to rate prospective meeting times. This is an improvement over binary scheduling, but is limited to five options, which must be manually ranked for each meeting.

Our Approach

Group scheduling is a complex task; there are certainly many other dimensions that could eventually be explored,

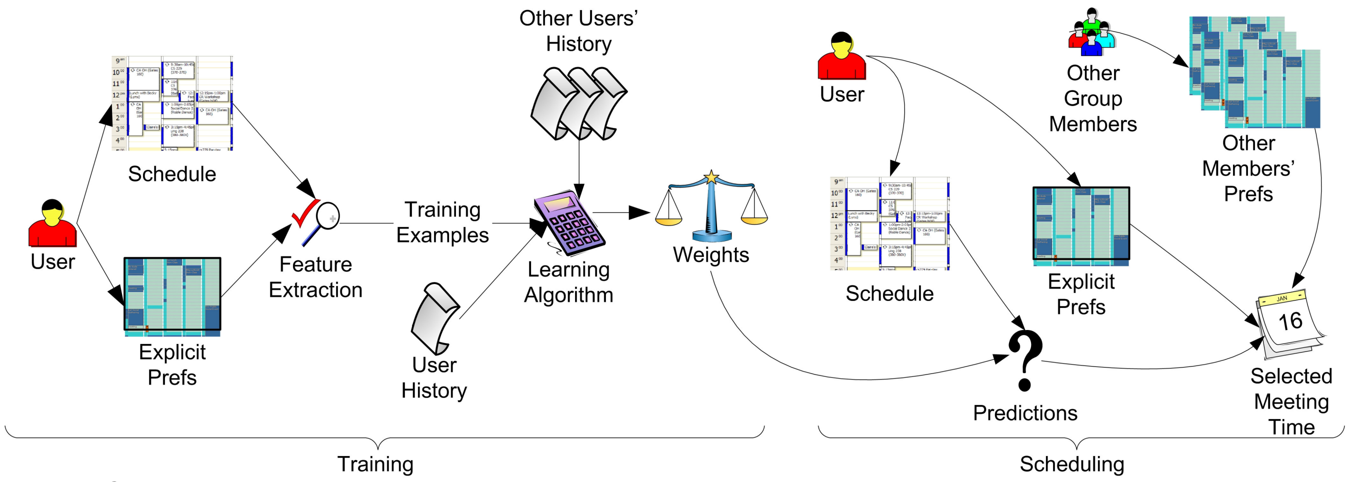


Figure 1 Our approach employs machine learning to train on users' past scheduling history and predict how they will respond.

such as event interdependencies, task deadlines, and flexibility. It could be considered a special case of activity synchronization, which has often been framed as a constraint satisfaction problem and is beyond the scope of this study. Similarly, there exist domains where scheduling does not occur in a 24×7 week. However, for the sake of study, we have reduced the problem to a simpler one of scheduling isolated meetings over the next week.

We focus on group *scheduling* rather than *calendaring*. We set out to investigate whether we could use a calendar to represent user *preferences* rather than availability, focusing only on a world-word mapping. This is important because:

· User preferences are more complex than binary "free/busy" availability. Some free times are more desirable than others, just as some busy times could be preempted under some circumstances.

· By detaching the portrayal of a user's preferences from his or her actual schedule, we preserve a user's privacy and afford him or her "plausible deniability" to prevent being scheduled for events against his or her will.

· By adding a layer of explicit user preferences, we no longer require users to maintain complete online schedules to gain value from such a system, potentially resolving the "prisoner's dilemma."

Of course, this task is much easier with face-to-face negotiation or when there are few parties to consider; we focus here on asynchronous group scheduling, because computer-mediated scheduling is often required.

Because it is time-consuming for a user to supply a complete set of preferences over all prospective times, we are interested in designing interactions that are as lightweight as possible. The specific method might vary depending on deployment context; for office users who already maintain digital schedule artifacts, this should likely be an extension of their current PIM. Our prototype does not require an existing PIM; building it as a standalone app enabled us to extend beyond traditional GCS users.

This paper introduces a machine learning approach that implicitly learns how users prefer to schedule time and then attempts to predict their responses. If successful, this reduces a user's interaction to correcting our system's guesses. However, users need to remain firmly in the loop: the reasoning behind these guesses should be exposed, and users must retain a sense of agency.

We seek to explore the confluence of visualization of, direct manipulation of, and machine learning on user preferences and their application to the group scheduling problem (see Figure 1). This paper will provide an overview of related work; describe current scheduling practices; and detail our preliminary prototype, efforts to build a machine learning model, and the resulting groupTime system.

RELATED WORK

This section provides a partial overview of prior work on the organizational and semantic study of calendaring, various attempts to automate scheduling, and the application of machine learning to assist in assessing users' availability.

Calendaring

Prior work highlights the difficulty of creating an effective GCS. Grudin [16] cites disparity between who is required to do additional work (employees maintaining their calendars) and who enjoys the benefits (managers who schedule meetings) and the resulting prisoner's dilemma problem as barriers to widespread use of groupware. He argues that challenges like this may explain why GS solutions tend to fail unless backed by a strong organizational force. We are interested in providing a system that is used and deemed beneficial without managerial enforcement.

Yet Palen [26] found that users are willing to share a great deal of information about their extended availability and schedules with colleagues if they gain better scheduling information and therefore can choose better meeting times. This encouraged us to explore higher-bandwidth means of expressing scheduling preferences.

An early model for this is Beard *et al.*'s Visual Scheduler [7], which introduced a "priority-based" scheduling system, allowing users to mark any given time slot with five levels of gray shading to indicate priorities. While they found this to be more efficient and reliable than manual scheduling, they made no attempt to prioritize free times.

Efforts have been made to formalize scheduling interaction as well. The Coordinator formalized each step of the negotiation by codifying acts like requests and commitments to automate the process [37]. However, socially it proved too rigorous for most users [33].

Intelligent Scheduling

While commercial systems such as Outlook and Notes stop short of scheduling meetings on a user's behalf, some research systems have explored automated scheduling.

Higa *et al.* compared an automated group scheduler to face-to-face and email coordination and found users were less satisfied with automatically selected times even though they actually resulted in fewer scheduling conflicts [17]. We believe this may be because their system did not expose its reasoning; it merely sent users a message with its answer.

Inspired by multi-agent systems, Sen *et al.* asked users to explicitly describe their preferences along eight dimensions, set thresholds, and assign them weights, configuring contract-based autonomous agents to negotiate on their behalves [31]. While this succeeded in selecting meeting times effectively, it remains to be seen whether "average" users are willing to tweak such a system directly; our approach is to learn these preferences implicitly.

The notion that people's scheduling interaction can be fully automated, that each person's desires can simply be distilled into an agent configured to act on his or her behalf, confuses, in Suchman's terms, plans and "situated actions" [34]. People will bend their own rules as the situation warrants—and often do to compromise. Rationality gives way to social pressures, as scheduling a meeting is inherently a social transaction [9]. So we have decided to keep users in control as much as possible.

While automated agents have been proposed as a general solution to information overload [24], direct manipulation beats out autonomous agents for some tasks [32]. Specifically, users like to feel in control of their schedules and tend to resist systems that deprive them of agency in the scheduling process [17, 26]. Our hybrid approach moves away from the fully-automated extreme, where the cost of mistakes is very high [21], and the fully manual extreme, where interaction is cumbersome.

Predicting User Behavior

Various different machine learning techniques have been successfully applied to a variety of HCI problems, including generating user interfaces [14], inferring structured activities from email [22], and mobile messaging [29]. Recent research has explored machine learning for predicting users' presence, interruptibility, and availability.

Dourish's Portholes let users make inferences about colleagues' presence and availability from direct observation [10], inspiring later implementations of desktop instant messaging such as [1] and [4]. Fogarty *et al.* took this a step further by using sensors such as keyboard and mouse activity and audio levels to build statistical models of interruptibility [12]. Horvitz *et al.* employed machine learning techniques to forecast users' short-term presence and predict their availability and interruptibility [18]. This work demonstrates that user availability patterns are predictable to some degree.

Tullio's Augur system sought to predict user behavior by using Bayesian networks to model whether users would actually attend scheduled meetings [36]. This sheds light on another crucial aspect of scheduling: the events on a user's calendar are not necessarily indicative of what he or she will do; in reality some scheduled events are preemptible—and which ones are to some extent predictable.

Gervasio *et al.* used support vector machines to learn user preferences for scheduling [15]. Their PLIANT system learns an ordering on pair-wise preferences by having users choose the best out of five options. We hope to obtain a richer dataset by obtaining absolute preferences over all possible meeting times rather than limiting the *universe of discourse* to five fixed meeting times.

CURRENT SCHEDULING PRACTICES

One of our goals is to extend group scheduling beyond traditional enterprise GCS users. We sought a user group who did not have access to a commercial GCS server: college students. We began with informal interviews with about 20 (primarily undergraduate) students from introductory HCI and communications courses.

While students may be relatively *inexperienced* at project management, they are certainly a community that could significantly *benefit* from improved tools: we found students to be heavily committed to a wide variety of obligations, and—unlike traditional office workers—are rarely collocated with group members. The students we interviewed have wildly varying and often chaotic schedules, with their commitments spanning a range of academic, social, extracurricular, and work-related activities. They schedule meetings with peers for a variety of purposes including: working on group projects, collaborating on problem sets, handling business for student organizations, rehearsing for an upcoming performance, and social gatherings. At our university, over 94% of undergraduates live on campus, making evenings and weekends at least as available and preferable as weekdays; in contrast, office workers ho usually find the workday more convenient for meetings.

Frequently, these meetings recur among the same defined set of people (*e.g.*, a study group or a committee). Often these are organized on an ad-hoc basis, rather than as a weekly commitment. One subject told us that her sorority is constantly faced with the challenge of finding a time that their 30 members can attend when they plan activities. This type of problem is typical of social groups that wish to stay connected for socializing or work. Such groups often find it difficult or unnecessary to commit to a regular meeting time each week, or occasionally need to schedule additional meetings.

Calendaring Artifacts

All students have some elements of their schedule that are part of a weekly routine. These include classes, rehearsals and practices, staff meetings, worship services, work, and volunteer commitments. Generally these schedules are solidified within the first two weeks of each academic quarter and don't change afterward.

Most of the students we interviewed create some artifact of their basic weekly schedules. This often takes the form of a spreadsheet or text document, entry in a desktop calendaring tool such as iCal [2] or Palm Desktop [6], or a paper schedule. It appears that students are willing to expend some effort at the beginning of each quarter to construct this artifact, but that is the extent of most students' interaction with calendaring tools. Why bother maintaining an online calendar if you can't take it with you in a more convenient form than your notebook computer?

Many students rely purely on memory to keep track of this week's schedule. Others use a variety of artifacts—scraps of paper, paper planners, and cell phones, since students are relatively mobile. While most students do not maintain a digital calendar, a minority use a PDA or a desktop PIM such as Outlook, if they can sync it with a portable device.

Scheduling Methods

Students' asynchronous scheduling primarily occurs via email. Hu and Brzozowski conducted a preliminary study [19] where they asked 20 students to schedule meetings with groups of four randomly selected members via email. They observed the exchanges that took place and debriefed participants on how they normally schedule meetings with other groups. Two basic strategies emerged:

Aggregation. One member acts as the "coordinator" (in our study we designated one from the group at random). The coordinator sends an email asking for a complete list of everyone's availabilities. Other members submit their free times, either in an email body or on a spreadsheet. The coordinator then manually combines everyone's free times and finds the intersection to choose a meeting time.

An advantage of this model is that the group has full disclosure of people's availabilities, each free slot on every participant's schedule being equally important. This ensures that all possibilities are considered. The chief downside is the investment of time required. Participants must enumerate every free time on their schedule, whether or not it is relevant, and the coordinator must take time to collect all the responses.

However, some groups favor this model because it shifts the burden onto one person, effectively absolving other members of responsibility. Aggregation creates a power dynamic whereby the organizer assumes control.

Negotiation. One member initiates an email thread by either enumerating a list of times he or she is available, or describing a few constraints (*e.g.*, "I'm free any day after 7"). Alternately, the discussion starts with a proposal ("How's Tuesday night?"). This informally sets the *universe of discourse* for the discussion, and other members of the group often focus on these blocks of time, even if there are others that work better for them. In successive rounds, the other members whittle down the universe of discourse by adding additional constraints, or expand it by making counteroffers. Some groups favor this model because it seems more democratic and requires no coordinator to step forward.

An advantage of this model is that if a group has compatible schedules they potentially don't have to discuss as many possibilities; if someone proposes a time that works well for everyone there is little debate.

A disadvantage is that expanding the universe of discourse is costly; if someone wishes to propose a new time, at least another full round of emails is required to gather everyone's assent. So it's easier to "go with the flow" and add constraints, making scheduling a sort of greedy search susceptible to local maxima rather than a true optimizing task.

In practice, it is common for groups to adopt a hybrid approach; for instance, a coordinator might assume authority but still open the floor for debate, or a group might aggregate their free times without an explicit coordinator.

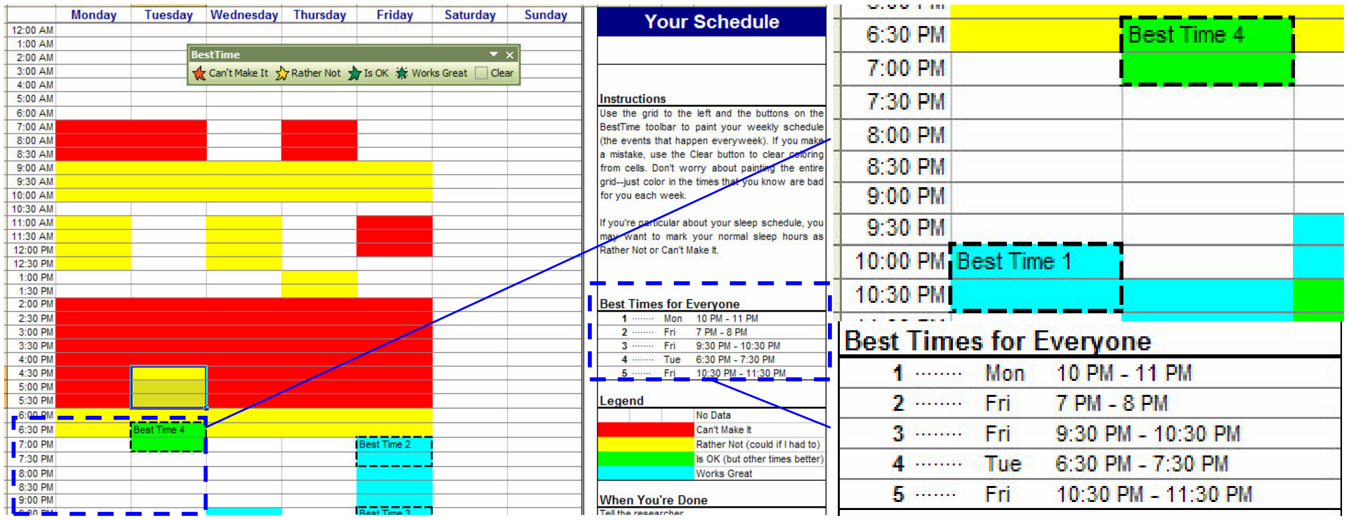
PRELIMINARY PROTOTYPE

For our first prototype, we sought to blend aggregation and negotiation in the context of scheduling a meeting to take place over the next week. It aggregates users' preferences across an entire week, and lets them negotiate by updating the best meeting times live in response to their feedback. This was intended as an early "proof-of-concept" prototype.

Implementation

Since a number of our users told us they use Excel to maintain a basic weekly schedule, we built a prototype in an Excel workbook (see Figure 2). Each sheet of the workbook has a grid representing the following week, with cells offering half-hour granularity (except for hour-long cells between midnight and 8AM).

Each cell can be marked with one of four weighted labels inspired by [11]—*Can't Make It* (0), *Rather Not* (1), *Is OK* (2), or *Works Great* (3)—by selecting a block of cells and clicking a tool from a floating toolbar. We chose a four-point rating scale because it offers more precision than a binary free/busy scheme, yet still forces users to take a



Your Schedule

Instructions

Use the grid to the left and the buttons on the BestTime toolbar to paint your weekly schedule (the events that happen everyweek). If you make a mistake, use the Clear button to clear coloring from cells. Don't worry about painting the entire grid--just color in the times that you know are bad for you each week.

If you're particular about your sleep schedule, you may want to mark your normal sleep hours as Rather Not or Can't Make It.

Best Times for Everyone

1 ········	Mon	10 PM - 11 PM
2 ········	Fri	7 PM - 8 PM
3 ········	Fri	9:30 PM - 10:30 PM
4 ········	Tue	6:30 PM - 7:30 PM
5 ········	Fri	10:30 PM - 11:30 PM

Legend

	No Data
🟥	Can't Make It
🟨	Rather Not (could if I had to)
🟩	Is OK (but other times better)
🟦	Works Great

When You're Done

Tell the researcher

Figure 2 *(Left)* An individual user's view of our first prototype; he or she can paint preferences onto the grid. *(Right)* The five best times to meet are outlined with dashed lines and listed in text, updated live in response to the user's changes.

stand: there is no neutral ground. Note that this model does not have any knowledge of a user's actual schedule; the user decides whether to mark a conflict as *Can't Make It*, *Rather Not*, or ignore it entirely.

Algorithm

Each labeling corresponds to a numeric value. Cells that the user has not colored in are assumed to be *OK* by default, since users who don't care enough to mark a cell *Can't Make It* or *Rather Not* probably don't mind meeting then. To determine the best time for a meeting, our system averages each user's rating for each cell to obtain its score. The best time to meet is then simply the block of consecutive cells of the desired meeting length with the lowest average score (weighted to compensate for the hour-long cells).

Simple averaging assumes each participant is equally important to the meeting but does not attempt to maximize the number of attendees; if a time is convenient for all but one member of a group, it takes the strict utilitarian view that that time is best. One may also envisage other algorithms.

Method

Twenty undergraduates from an introductory communication class participated in our user study. The 12 men and 8 women were randomly assigned to five groups of four students.

As with our structured interviews, we asked each group to use email to find two possible times for what we described as a "focus group." Upon reaching consensus, we asked representatives to forward all their scheduling mail to us. We weren't sure whether groups adopting the aggregation style would "reply to all" or just direct messages to their coordinator, so we wanted to preserve this behavior rather than give them mailing lists so we could observe *in situ*.

At the meeting they scheduled, we asked members of each group to take turns using our prototype to schedule an hour-long meeting for the coming week. Due to compatibility issues with Excel, we were not able to deploy it to "the wild" so we simulated an asynchronous scheduling process by having users complete these tasks in rounds:

1 Starting with a blank spreadsheet, color in the cells corresponding to your basic quarterly schedule (*i.e.*, the commitments that do not change week-to-week).

2 Starting with the spreadsheet you colored previously, now with the best times to meet outlined, recolor cells "until you're satisfied with the meeting times."

3 (as necessary) Revise your ratings in response to your group's responses.

Finally, the participants filled out an online questionnaire to compare this scheduling process with email.

Results

Over email, our users all elected a *negotiation* strategy rather than *aggregation*. Each participant read an average of seven messages before a meeting time was selected. Overall, users were comfortable with the notion of expressing absolute preferences for times as in our prototype. We expected users to color in the bare minimum of cells necessary to obtain the outcome required but a number of them painted in every cell of the week even though we told them they didn't have to. These users were less likely to have to revise their ratings on a second pass.

We also observed that users were surprisingly cooperative. Several times people who initially described themselves as busy most of the week actually reconsidered once they saw that the best times for everyone else were times they had said they *Can't Make It* or would *Rather Not*. The visibility of group members' availability exerts a form of social pressure to encourage compromise and honesty.

We also saw participants use the *Works Great* label as a tool to suggest alternate times. Since *Works Great* has a higher value than the default, the system is likely to call attention to it as one of the best times, unless other users rated it a bad time.

Marking the best times on the schedule provided live feedback to users; if the "best time" on the schedule was one that didn't work for them they would label it *Can't Make It* or *Rather Not*. If it was still the "best time", the only way to change it would be to propose a better time by marking it *Works Great*, or to elevate one of the other candidate "best times" by rating them *Is OK* or *Works Great*.

Our users generally indicated they found our prototype "easy to use" and would be likely to use an application based on it and to recommend it to friends. However, this prototype was limited in scope and did not reflect "real-world" use conditions, as the users were collocated to use our prototype *after* they met each other [28]. So this result should be considered circumstantial.

LEARNING PREFERENCES

Having seen what we could do without any knowledge of actual user schedules (beyond users' ratings) we wondered: what if we *did* know about their commitments? Could we infer users' preferences from looking at their schedules and past behavior? In this case, we'd be able to "paint" the entire schedule with our guesses, and reduce the interaction to correcting the system's guesses. We set out to build a model of users' scheduling behavior to do just that: predict how a user would respond to any given meeting request.

At its most basic level, scheduling is a constraint satisfaction problem. One approach is to ask users to explicitly declare the rules governing their preferences, as in [31]. However, this assumes that scheduling decisions are purely rational and that users can easily explain their reasoning. Interviewing our subjects made it clear that a wide variety of factors affect user preferences, not all of which could be easily elucidated. For instance, subjects had difficulty devising a rank ordering of their commitments' importance. Enumerating a complete set of rules is also rather laborious for users. For these reasons, a simple rule-based engine seems a poor model for user behavior; we employ a probabilistic model that learns preferences implicitly, and can make predictions along with a confidence metric.

Implementation

Based on our interviews with students we constructed a basic ontology of how people prefer to schedule meetings.

Scheduling Ontology

There are two separate but related questions at play here: how people deal with scheduling conflicts and how people prefer to schedule their "free" time. The answers vary widely from person to person but we hope to distill out some common decision factors.

To address the first, we considered that users somehow prioritize the events on their calendar and their commitments. People will skip or reschedule an appointment or meeting if something more important comes along. Codifying that precisely is difficult, however, since people are not entirely rational in social interactions [9]. Rather than seek explicit prioritization from users, we wanted to get a sense of this priority scheme implicitly.

We sought to capture generically the *nature* and *degree of obligation* a user feels to attend an event on his or her calendar. Our subjects' schedules reflect a variety of commitments competing for their time: academic, extracurricular, economic, personal, and social. Within each type of commitment, we tried to capture varying costs of missing a scheduled event. Often this comes down to *who* gets hurt by an absence and how serious the repercussions are.

After much discussion with students, we settled on the event *types* shown in Table 1. This schema accounts for the subtle differences, for example, between a frat party where the user's absence won't be appreciably noticed and a date, for which the consequences of not attending are much greater. We chose these factors:

Conflicts. If a meeting would conflict with a scheduled event, the type of event and whether it's a recurring or one-time-only event.

Day and time. Some people are "morning people"; others

	Type	Description	Costs of missing
Academic	Lecture	Attendance-optional lecture for a class	Primary course material
	Project	Meeting to work on a team project for a grade	Letting down a team; grade
	Section	Optional discussion section for a class	Supplementary course material
	Seminar	Mandatory class session	Grade; supplementary material
	Study, group	Group study session (presence not required)	Indirect effect on grade
Activities	Rehearsal	Rehearsal or athletic practice (usually mandatory)	Letting down a team/group
	Meeting	Non-class-related meeting	Letting down a group
Economic	Interview	A (difficult to obtain) job interview	Career opportunities
	Work	Work as part of a paying job	Wages; may get fired
Personal	Interest	An optional event of personal interest (not with friends)	Self-edification
	Study, alone	Planned study time (flexible)	Grade
	Sleep	Planned sleep time (flexible)	Health
Social	Social, private	Small social event with one or more friends	Letting down friends
	Social, public	Large social event (no one will notice user's absence)	Opportunity to meet people

Table 1 Our ontology of college students' commitments captures a wide range of obligations, allowing us to learn users' priorities implicitly rather than asking for an explicit ranking.

prefer to meet later at night. This also gives us the ability to infer when a user is *usually* busy even if he or she chooses not to explicitly tell us of a commitment.

Adjacent events. Some people prefer to have large chunks of free time in their schedules and stack events up back-to-back; others prefer to space their commitments out more. Additionally, there are special cases for some types of prior or succeeding events; for instance, right before a job interview or right after a study group meeting.

Type of meeting. People are more flexible (or more willing to sacrifice) depending on the type of meeting.

Note that this ontology is designed for its user group, based on our users. Our approach could be adapted for other populations with further study to develop additional ontologies. For instance, Tullio demonstrated an ontology successful at predicting behavior in an office setting [36]. There may not be a universal ontology for all users, but different models could be used for different groups.

Algorithm

For each potential meeting time, our system extracts schedule-agnostic *features* from each user's calendar using our ontology. This enables us to compare behavior from one situation to another without relying on the original schedule. It uses softmax regression [25] (a generalization of logistic regression) to assign a set of *weights* to each feature that can be used to predict the rating of a potential meeting time.

Other machine learning algorithms, such as support vector machines [8] and Bayesian networks [13], can also be applied to this task. We chose softmax regression mainly for the sake of simplicity, but also because it makes predictions by assigning weights to each feature, and thus its output can be intuitively explained to users, adding introspectability.

To derive weights, our algorithm extracts features from a user's calendar and creates a set of *training examples* for each potential meeting time that he or she has rated. To predict a user response to a new training example, our algorithm takes the dot product of the weights for each rating *j*, θ_j, and the features of a given training example. Using softmax, the probability that the user will rate a meeting with features x with rating i given weights θ is [25]:

$$p(i|x;\theta) = \frac{\exp(\theta_i^T x)}{\sum_j \exp(\theta_j^T x)}$$

The prediction is the rating i that maximizes this expression, and the confidence is its probability. Each feature contributes some factor to this value that is determined by the weights.

Gathering Schedule Data

We sought to validate our model and see whether it can learn user preferences based on their schedules. We solicited users' complete schedules for the following week. Users were asked to categorize each of their commitments according to the types we defined in Table 1, and indicate whether they were recurring or one-time-only events. We then presented a series of 40 hypothetical hour-long meetings and asked them to rate five different meeting time options for each.

We ran our feature extraction algorithm on their responses and calendars to generate 200 examples. For each user, we evaluated our algorithm's ability to learn their preferences by averaging over ten random 70/30 train/test splits; that is, for each user we ran ten trials where we trained on a randomly selected 70% of his or her examples and then tested on the remaining 30% by comparing our prediction to the user's actual response.

Results

Forty-six students completed our experiment. While results varied widely, for a third of the users the system predicted the correct rating at least 70% of the time; for half of them it predicted the correct rating at least 62% of the time. More frequently, our model could guess either the correct rating or one rating adjacent (*e.g.*, *Is OK* instead of *Works Great*). For half the users, our model was either correct or one rating off 90% or more of the time.

Users whose predictions were more error-prone (at least two standard deviations above the mean) all had provided us with relatively sparse calendar data, indicating the model works better with more knowledge of a user's schedule.

We also attempted to generate a basic set of weights for all users, to help us make predictions for users who have never provided us with training examples. Some examples of these features are shown in Figure 3.

GROUPTIME SYSTEM

Armed with these results, we created the complete Web-based groupTime prototype to explore the utility of partial sharing. As it was Web-based, people could use our prototype asynchronously outside of a lab setting. We also wanted to try to democratize the process as much as possible, so that no voice in a group outweighs any other.

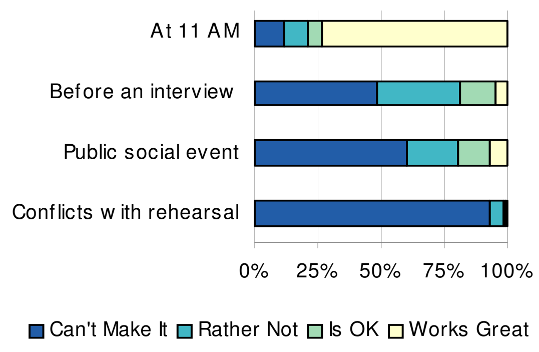
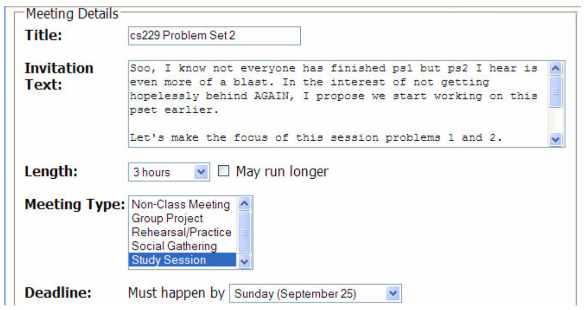


Figure 3 A few examples of features and the probability distributions they imply. Some features are strong indicators of a particular rating, while others are weaker.



Meeting Details

Title: cs229 Problem Set 2

Invitation Text:

```
Soo, I know not everyone has finished ps1 but ps2 I hear is
even more of a blast. In the interest of not getting
hopelessly behind AGAIN, I propose we start working on this
pset earlier.

Let's make the focus of this session problems 1 and 2.
```

Length: 3 hours ☐ May run longer

Meeting Type: Non-Class Meeting / Group Project / Rehearsal/Practice / Social Gathering / [✓ Study Session]

Deadline: Must happen by Sunday (September 25)

Figure 4 The organizer of the meeting provides a general description and a deadline.

Implementation

groupTime is a Web application, powered by servlets. Users interact with Web pages and a Java applet to respond to meeting invitations. Much like setting up a meeting in Evite or Outlook, someone starts the process by scheduling a new meeting, specifying the meeting's type, length, and by when it must happen (see Figure 4). However, the organizer does not get to specify any meeting time explicitly.

Each guest receives an email inviting him or her to the meeting, with a link to respond. Each user, including the organizer, uses the applet shown in Figure 5 to indicate his or her preferences by selecting one of the color-coded preferences tools and "painting" them onto the grid. The organizer has the added power to veto blocks of time he or she deems not to be up for discussion, for instance late nights for an office meeting or daytime for a movie night.

Users have the opportunity to add as many or as few events from their schedules as they wish; events added by a user are stored for any other meetings he or she is involved in. Using the default weights and any events the user adds, we dynamically predict ratings for them. The system can use any weights that we wish, generated by any learning algorithm, allowing it to update the user's weights as it trains on his or her responses. For predictions where it doesn't have at least 75% confidence, it falls back to a binary classification task: if it's more likely that the user can't make it *or* would rather not, it guesses *Rather Not*; otherwise it guesses *Is OK*. This smoothes out fluctuations generated by our model when it has too little information.

We faced a critical design challenge in that groupTime produces multivariate data. For any given time we have a labeling, a confidence (or a user's explicit choice), and responses from other users. This creates a tension between conveying as much information as possible and making interaction lightweight and intuitive. We chose to present only two dimensions of data, user labeling and group aggregate (a strip along the left), to make the interaction more lightweight. In more mission-critical decision tools, a designer may choose to expose additional axes of data.

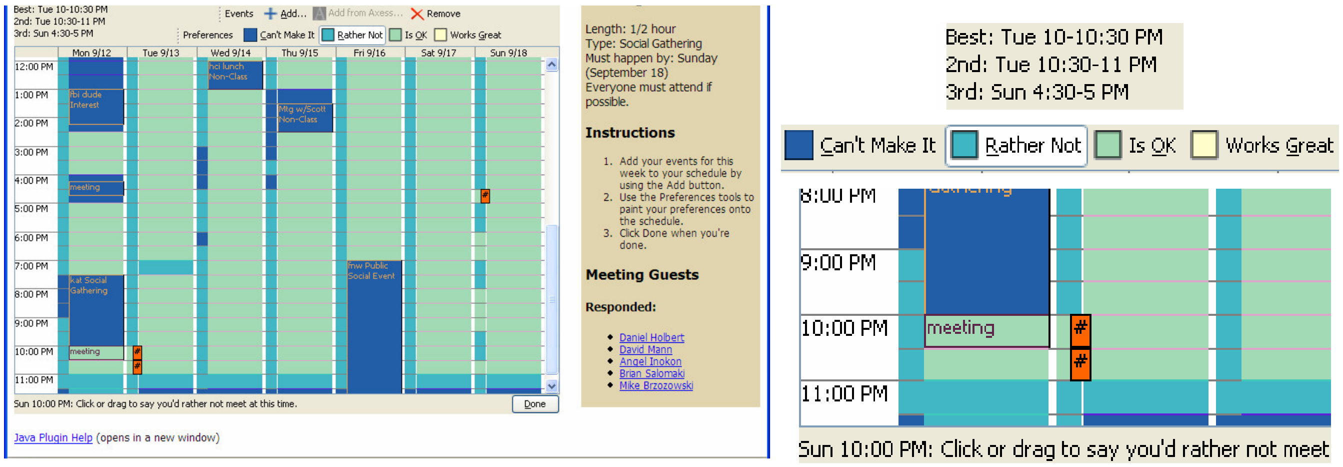
Our classifier will no longer change any cell a user has rated, even if other events are added, to avoid appearing to change things behind his or her back. All cells have some guesses, and users have an opportunity to correct those guesses by rating cells. Once a user clicks Done, the system takes all ratings, whether guessed or explicitly stated, as the user's preferences, and does not change them even in the face of new information.

In addition to the utilitarian algorithm used in our preliminary prototype, we added an algorithm that attempts to maximize the number of attendees (perhaps at the expense of someone's happiness). Meeting times are ordered by how many people can attend and then by the average rating.

Once everyone has responded, the system emails everyone with the best time according to everyone's preferences. If anyone is unhappy with the time chosen, he or she can click a link in this message to return to the response page and update his or her preferences, effectively renegotiating the meeting time. Each message has the semantic meaning of "unless there are any objections, we're meeting then"; each participant has the ability to object right until the start of the meeting.

Methodology

We recruited 35 volunteers from the Stanford community to participate in a 30-minute user study in groups. Users were



Best: Tue 10-10:30 PM
2nd: Tue 10:30-11 PM
3rd: Sun 4:30-5 PM

Sun 10:00 PM: Click or drag to say you'd rather not meet at this time.

Java Plugin Help (opens in a new window)

Length: 1/2 hour
Type: Social Gathering
Must happen by: Sunday (September 18)
Everyone must attend if possible.

Instructions

1. Add your events for this week to your schedule by using the Add button.
2. Use the Preferences tools to paint your preferences onto the schedule.
3. Click Done when you're done.

Meeting Guests

Responded:

- Daniel Holbert
- David Mann
- Angel Inokon
- Brian Salomaki
- Mike Brzozowski

Figure 5 Responding to an invitation with our second prototype. The left strip in each column shows the aggregate of the group's preferences. Like our first prototype, groupTime provides live feedback of the best times to meet (marked with #s).

encouraged to sign up for the study in groups with friends or colleagues when possible to exploit the social dynamics of pre-existing groups who already get together regularly.

Our groups included circles of friends, residential computer consultants from the same region, and a workgroup, as well as three groups of randomly assigned people. We added one group of office workers to see how expectations differ in a business context. Our subjects consisted of approximately 26% undergraduates, 52% graduate students, and 22% office workers. About 68% of participants scheduled meetings with groups of 3 to 4 people at least once a week. Groups ranged from three to five members in size.

We divided subjects up into control and experimental sets. The control set was asked to schedule their user study entirely by email, while the experimental set used our new prototype. We told everyone that all members of their group must be present at the user study to receive compensation. We appointed a representative from each group, charged with reporting the desired meeting time back to us; representatives were the subjects who signed up their group and were randomly chosen in the unaffiliated groups.

Our preliminary study found virtually all email traffic went to the whole group, so we gave each control set group a mailing list to use, allowing us to monitor their discussions. At the user study, groups in the control set were shown a demonstration of the prototype and had the opportunity to ask questions. Subjects completed a questionnaire pertaining to their scheduling experience (whether email or our prototype) and participated in a focus group discussion about their scheduling experience and the prototype system.

Results

Subjects showed great enthusiasm for our approach to preference-based scheduling, particularly in comparison to email, but raised implementation issues that help us better understand desirable attributes of intelligent user interfaces.

Attendance

Of the 35 volunteers who signed up, 25 committed to attending a meeting. Ten people did not agree to a specific meeting time. One control group failed to find a meeting time because its representative did not contact his group. Another control group member tried to spur discussion over email but without decisive leadership, they never chose a specific time. Several experimental groups had one member who never responded to group emails or our system's invitation; after attempts to contact them, we dropped them from their respective groups. All but two of the 25 subjects showed up at their meetings: one subject misread the time in his email; another was sick on the day of the meeting.

User Reactions

Subjects were more likely to say email scheduling "requires many rounds of negotiation" than our prototype ($p = .05$). "I know that I don't have to deal with 15 minutes of 'How about Friday? I can't I'm busy then…," remarked one. With our prototype users were slightly more likely to say they "could influence the outcome," though not statistically significantly so ($p = .22$). One user remarked, "It takes out the social engineering of forceful personalities domineering a meeting to their own ends."

Subjects expressed a desire in particular to use our prototype to schedule meetings with large groups and committees that don't have frequent contact. In general, subjects said they were most likely to use our prototype to schedule meetings with groups of five or more people. While 83% of subjects said they were likely to use our prototype to schedule a "group project meeting," only 36% were comfortable using it to plan a "social gathering."

Democratic Interaction

People liked the egalitarian nature of the groupTime algorithm; one told us, "I like how the choice of timing is kind of out of any one person's hands, which makes it more likely to be what's best for the group as a whole." However, pure democracy can break down. One group in the experimental set had difficulty because one member had his schedule change 20 minutes before the appointed meeting time. He logged in to register his changes but it was too late; half of his group never got his email. Several people expressed a desire to have the coordinator or the system "lock down" a meeting time some time before the actual meeting, to make it seem "firm."

Others felt disconnected from the pulse of their group because our prototype doesn't disclose how individual members responded. Users wanted to know *why* cells were colored the way they were, particularly when someone else was responsible. Some felt compelled to explain themselves and wanted the ability to post comments to the group. Some people had trouble because they weren't able to check their email regularly, and suggested sending SMS messages to their cell phones to remind them.

In general, groupTime yielded successful meetings. The study also indicated two important design considerations. First, users desire more structure and slightly more disclosure than we previously thought. Second, most subjects agreed that groupTime should tip the balance of power in favor of the organizer more.

CONCLUSION AND FUTURE WORK

We demonstrated the viability of using preferences rather than explicit calendar sharing to schedule group meetings, but showed that calendars still provide valuable information that can be used to predict users' preferences. We also present four design considerations for semi-automated GS:

1 *Let the user know what's going on*. Users lose some agency when they let machines negotiate for them. Whether making a prediction based on past behavior or making a decision, the user should have some easy way to see *why* the system did what it did.

2 *Don't make changes behind the user's back.* Users prefer to have veto power over an agent and expect their decisions to be final. A system should not override users' explicit preferences, even in the face of new information, nor should it schedule a meeting without user input.

3 *Users seek predictability.* A system should always make it clear if and when a meeting is scheduled. In intelligent user interfaces, optimal results are desirable but a sense of predictability and consistency are also important [32].

4 *Use social pressure.* To avoid having one user "game the system" and appear deceitfully inflexible, let users see who the culprit is *and* let them explain themselves. Users who genuinely are busier than their colleagues shouldn't have to feel guilty if they have good reason, nor should cheaters get away with it.

While this methodology shows promise, more extensive study is required to validate this system in a larger context. We are currently preparing a longer-term deployment and a longitudinal study to ascertain how well we can adapt to individual users' unique priorities over time.

ACKNOWLEDGMENTS

We thank the participants of our three studies and members of the Stanford HCI Group for their feedback. We also thank Tom Dietterich, Melinda Gervasio, and Leslie Kaelbling for helpful conversations. Brzozowski and Carattini were partially supported by the U.S. Department of the Interior/DARPA under contract number NBCHD030010. Intel donated hardware for our server and our user studies. Human subject research was governed by Stanford University IRB-approved protocols 902 and 3439.

REFERENCES

1 *AOL Instant Messenger*, 2005. America Online, Inc.: Dulles, VA. www.aim.com

2 *iCal*, 2005. Apple Computer, Inc.: Cupertino, CA. www.apple.com/macosx/features/ical/

3 *Lotus Notes*, 2005. IBM Corporation: Armonk, NY. www.ibm.com/software/lotus/

4 *MSN Messenger*, 2005. Microsoft Corp.: Redmond, WA. messenger.msn.com/

5 *Outlook*, 2003. Microsoft Corp.: Redmond, WA. office.microsoft.com

6 *Palm Desktop*, 2004. PalmSource, Inc.: Sunnyvale, CA. www.palmos.com/dev/tools/desktop/

7 Beard, D., M. Palaniappan, A. Humm*, et al.* A visual calendar for scheduling group meetings. *Proc. of CSCW 1990*: ACM Press. pp. 279-90, 1990.

8 Boser, B. E., I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. *Proc. Workshop on Computational Learning Theory*: ACM Press. pp. 144-52, 1992.

9 Dourish, P., Social Computing, in *Where the Action Is: The Foundations of Embodied Interaction*. MIT Press: Cambridge, MA. pp. 55-98, 2001.

10 Dourish, P. and S. Bly. Portholes: supporting awareness in a distributed work group. *Proc. CHI 1992*: ACM Press. pp. 541-47, 1992.

11 Evite, www.evite.com. 2005.

12 Fogarty, J., S. E. Hudson, and J. Lai. Examining the robustness of sensor-based statistical models of human interruptibility. *Proc. CHI 2004*: ACM Press. pp. 207-14

13 Friedman, N., D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning* **29**(2-3). pp. 131-63, 1997.

14 Gajos, K. and D. S. Weld. SUPPLE: automatically generating user interfaces. *Proc. IUI 2004*: ACM Press. pp. 93-100, 2004.

15 Gervasio, M. T., M. D. Moffitt, M. E. Pollack*, et al.* Active preference learning for personalized calendar scheduling assistance. In Proc. *IUI 2005*: ACM Press. pp. 90-97, 2005.

16 Grudin, J. Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM* **37**(1), 1994.

17 Higa, K., B. Shin, and V. Sivakumar. Meeting Scheduling: An Experimental Investigation. In Proc. *IEEE International Conference on Systems, Man., and Cybernetics*, 1996.

18 Horvitz, E., P. Koch, C. M. Kadie, and A. Jacobs. Coordinate: Probabilistic Forecasting of Presence and Availability. In Proc. *Conf. on Uncertainty and AI*: Morgan Kaufmann, 2002.

19 Hu, J. and M. Brzozowski. Preference-Based Group Scheduling. In Proc. *Interact 2005*, 2005.

20 Hutchins, E., *Cognition in the Wild*. Cambridge, MA: MIT Press 1995.

21 Isbell, C. L. and J. S. Pierce. An IP Continuum for Adaptive Interface Design. In Proc. *HCI International*, 2005.

22 Kushmerick, N. and T. Lau, Automated email activity management: an unsupervised learning approach, in *IUI 2005*. 2005, ACM Press: San Diego, CA.

23 Lamming, M., P. Brown, K. Carter*, et al.* The Design of a Human Memory Prosthesis. *The Computer Journal* **37**(3). pp. 153-63, 1994.

24 Maes, P. Agents that reduce work and information overload. *Commun. ACM* **37**(7). pp. 30-40, 1994.

25 McCullagh, P. and J. A. Nelder, *Generalized Linear Models*. 2nd ed. London: Chapman and Hall 1989.

26 Palen, L., *Calendars on the New Frontier: Challenges of Groupware Technology*, University of California, Irvine, 1998.

27 Palen, L. Social, individual and technological issues for groupware calendar systems. *Proc. CHI 1999*: ACM Press. pp. 17-24, 1999.

28 Rocco, E., Trust breaks down in electronic contexts but can be repaired by some initial face-to-face contact, in *CHI 1998*. 1998, ACM Press/Addison-Wesley.

29 Schmandt, C., N. Marmasse, S. Marti*, et al.* Everywhere messaging. *IBM Syst. J.* **39**(3-4). pp. 660-77, 2000.

30 Searle, J., *Speech acts: an essay in the philosophy of language*. London: Cambridge University Press 1969.

31 Sen, S., T. Haynes, and N. Arora. Satisfying user preferences while negotiating meetings. *Int. J. of Human-Computer Studies* **47**(3). pp. 407-27, 1997.

32 Shneiderman, B. and P. Maes. Direct manipulation vs. interface agents. *interactions* **4**(6). pp. 42-61, 1997.

33 Suchman, L. Speech acts and voices: response to Winograd et al. *Comput. Supported Coop. Work* **3**(1). pp. 85-95, 1995.

34 Suchman, L. A., *Plans and Situated Actions: The Problem of Human-Machine Communication*: Cambridge University Press. 203 1987.

35 Thrift, N., The Making of a Capitalist Time Consciousness, in *The Sociology of Time*, J. Hassard, Editor. St. Martin's Press: New York, 1990.

36 Tullio, J. Intelligent groupware to support communication and persona management. *Proc. UIST 2003*, 2003.

37 Winograd, T. A Language/Action Perspective on the Design of Cooperative Work. *HCI* **3**(1). pp. 3-30, 1988.